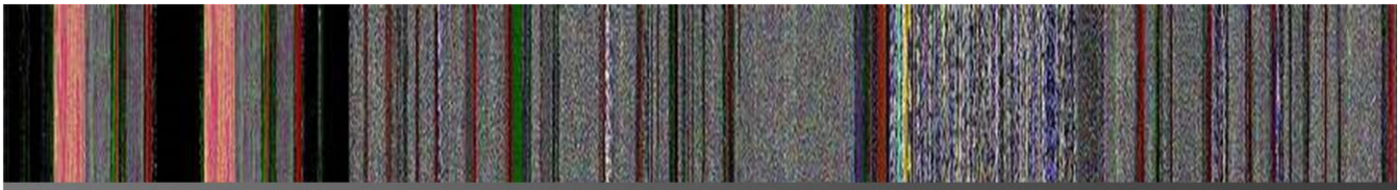


Defect

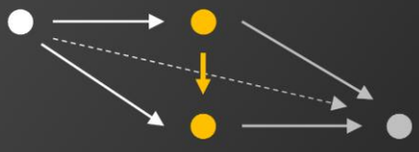
Detect

Windows API

for Software Diagnostics

Accelerated

With Category Theory in View



Dmitry Vostokov
Software Diagnostics Services

Published by OpenTask, Republic of Ireland

OpenTask books and magazines are available through booksellers and distributors worldwide. For further information or comments, send requests to press@opentask.com.

A CIP catalog record for this book is available from the British Library.

ISBN-l3: 978-1-912636-63-1 (Paperback)

Revision 1.00 (December 2022)

Contents

Exercise W1

Goal: Compare calling conventions on x86 and x64 platforms.

ADDR Patterns: Call Prologue; Call Parameter; Function Prologue.

1. Let's look at a 64-bit process. Launch WinDbg Preview.

2. Open \AWAPI-Dumps\Process\wordpad.DMP

3. We get the dump file loaded:

```
Microsoft (R) Windows Debugger Version 10.0.25200.1003 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\AWAPI-Dumps\Process\wordpad.DMP]
User Mini Dump File with Full Memory: Only application data is available


************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Windows 10 Version 22000 MP (2 procs) Free x64
Product: WinNt, suite: SingleUserTS Personal
Edition build lab: 22000.1.amd64fre.co_release.210604-1628
Machine Name:
Debug session time: Sat Oct 15 20:19:05.000 2022 (UTC + 0:00)
System Uptime: 0 days 0:27:57.202
Process Uptime: 0 days 0:00:26.000
.............................................................
.....
Loading unloaded module list
.
For analysis of this file, run !analyze -v
win32u!NtUserGetMessage+0x14:
00007ffc`f3811414 c3              ret
```

4. Open a log file using **.logopen**:

```
0:000> .logopen C:\AWAPI-Dumps\W1.log
Opened log file 'C:\AWAPI-Dumps\W1.log'
```

5. There are several places in Wordpad code where windows are created. We look at two of them. First, disassemble the *InitMainThreadWnd* function from the *combase* module and find the *CreateWindowExW* API call:

```
0:000> uf combase!InitMainThreadWnd
combase!InitMainThreadWnd [onecore\com\combase\object\mainthrd.cxx @ 114]:
  114 00007ffc`f5e14770 4c8bdc          mov     r11,rsp
  114 00007ffc`f5e14773 53              push    rbx
  114 00007ffc`f5e14774 4883ec60        sub     rsp,60h
  148 00007ffc`f5e14778 498363f000      and     qword ptr [r11-10h],0
  148 00007ffc`f5e1477d 4c8d05b45e1d00  lea     r8,[combase!`string' (00007ffc`f5fea638)]
  148 00007ffc`f5e14784 488b059d1a2400  mov     rax,qword ptr [combase!g_hinst (00007ffc`f6056228)]
  148 00007ffc`f5e1478b 41b900000088    mov     r9d,88000000h
```

```
  148 00007ffc`f5e14791 488b15c0232400  mov     rdx,qword ptr [combase!gOleWindowClass (00007ffc`f6056b58)]
  148 00007ffc`f5e14798 33c9            xor     ecx,ecx
  148 00007ffc`f5e1479a 498943e8        mov     qword ptr [r11-18h],rax
  148 00007ffc`f5e1479e b800000080      mov     eax,80000000h
  148 00007ffc`f5e147a3 498363e000      and     qword ptr [r11-20h],0
  148 00007ffc`f5e147a8 49c743d8fdffffff mov     qword ptr [r11-28h],0FFFFFFFFFFFFFFFDh
  148 00007ffc`f5e147b0 89442438        mov     dword ptr [rsp+38h],eax
  148 00007ffc`f5e147b4 89442430        mov     dword ptr [rsp+30h],eax
  148 00007ffc`f5e147b8 89442428        mov     dword ptr [rsp+28h],eax
  148 00007ffc`f5e147bc 89442420        mov     dword ptr [rsp+20h],eax
  148 00007ffc`f5e147c0 48ff15411e2700  call    qword ptr [combase!_imp_CreateWindowExW (00007ffc`f6086608)]
  148 00007ffc`f5e147c7 0f1f440000      nop     dword ptr [rax+rax]
  148 00007ffc`f5e147cc 4889057d232400  mov     qword ptr [combase!ghwndOleMainThread (00007ffc`f6056b50)],rax
  163 00007ffc`f5e147d3 4885c0          test    rax,rax
  163 00007ffc`f5e147d6 0f84e8950b00    je      combase!InitMainThreadWnd+0xb9654 (00007ffc`f5ecddc4)  Branch

combase!InitMainThreadWnd+0x6c [onecore\com\combase\object\mainthrd.cxx @ 180]:
  180 00007ffc`f5e147dc 48ff15fd081c00  call    qword ptr [combase!_imp_GetCurrentThreadId (00007ffc`f5fd50e0)]
  180 00007ffc`f5e147e3 0f1f440000      nop     dword ptr [rax+rax]
  180 00007ffc`f5e147e8 890572232400    mov     dword ptr [combase!gdwMainThreadId (00007ffc`f6056b60)],eax
  181 00007ffc`f5e147ee 33db            xor     ebx,ebx

combase!InitMainThreadWnd+0x80 [onecore\com\combase\object\mainthrd.cxx @ 186]:
  186 00007ffc`f5e147f0 8bc3            mov     eax,ebx
  187 00007ffc`f5e147f2 4883c460        add     rsp,60h
  187 00007ffc`f5e147f6 5b              pop     rbx
  187 00007ffc`f5e147f7 c3              ret

combase!InitMainThreadWnd+0xb9654 [onecore\com\combase\object\mainthrd.cxx @ 168]:
  168 00007ffc`f5ecddc4 48ff1525701000  call    qword ptr [combase!_imp_GetLastError (00007ffc`f5fd4df0)]
  168 00007ffc`f5ecddcb 0f1f440000      nop     dword ptr [rax+rax]
  168 00007ffc`f5ecddd0 8bd8            mov     ebx,eax
  168 00007ffc`f5ecddd2 85c0            test    eax,eax
  168 00007ffc`f5ecddd4 7e09            jle     combase!InitMainThreadWnd+0xb966f (00007ffc`f5ecdddf)  Branch

combase!InitMainThreadWnd+0xb9666 [onecore\com\combase\object\mainthrd.cxx @ 168]:
  168 00007ffc`f5ecddd6 0fb7d8          movzx   ebx,ax
  168 00007ffc`f5ecddd9 81cb00000780    or      ebx,80070000h

combase!InitMainThreadWnd+0xb966f [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 00007ffc`f5ecdddf 833d0265180000  cmp     dword ptr [combase!_Tlgg_hCombaseTraceLoggingProviderProv
(00007ffc`f60542e8)],0
  169 00007ffc`f5ecdde6 7714            ja      combase!InitMainThreadWnd+0xb968c (00007ffc`f5ecddfc)  Branch

combase!InitMainThreadWnd+0xb9678 [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 00007ffc`f5ecdde8 833dbd85180000  cmp     dword ptr [combase!gfEnableTracing (00007ffc`f60563ac)],0
  169 00007ffc`f5ecddef 743b            je      combase!InitMainThreadWnd+0xb96bc (00007ffc`f5ecde2c)  Branch

combase!InitMainThreadWnd+0xb9681 [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 00007ffc`f5ecddf1 33c9            xor     ecx,ecx
  169 00007ffc`f5ecddf3 e8b4ecedff      call    combase!IsWppLevelEnabled (00007ffc`f5dacaac)
  169 00007ffc`f5ecddf8 84c0            test    al,al
  169 00007ffc`f5ecddfa 7430            je      combase!InitMainThreadWnd+0xb96bc (00007ffc`f5ecde2c)  Branch

combase!InitMainThreadWnd+0xb968c [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 00007ffc`f5ecddfc 488d05c5b91100  lea     rax,[combase!`string' (00007ffc`f5fe97c8)]
  169 00007ffc`f5ecde03 895c2430        mov     dword ptr [rsp+30h],ebx
  169 00007ffc`f5ecde07 4889442428      mov     qword ptr [rsp+28h],rax
  169 00007ffc`f5ecde0c 4c8d055d7b1400  lea     r8,[combase!`string' (00007ffc`f6015970)]
  169 00007ffc`f5ecde13 8364242000      and     dword ptr [rsp+20h],0
  169 00007ffc`f5ecde18 488d15297b1400  lea     rdx,[combase!`string' (00007ffc`f6015948)]
  169 00007ffc`f5ecde1f 41b9a9000000    mov     r9d,0A9h
  169 00007ffc`f5ecde25 8bcb            mov     ecx,ebx
  169 00007ffc`f5ecde27 e8ec9aeaff      call    combase!ComTraceMessage (00007ffc`f5d77918)

combase!InitMainThreadWnd+0xb96bc [onecore\com\combase\object\mainthrd.cxx @ 170]:
  170 00007ffc`f5ecde2c 85db            test    ebx,ebx
  170 00007ffc`f5ecde2e 0f88bc69f4ff    js      combase!InitMainThreadWnd+0x80 (00007ffc`f5e147f0)  Branch

combase!InitMainThreadWnd+0xb96c4 [onecore\com\combase\object\mainthrd.cxx @ 172]:
  172 00007ffc`f5ecde34 bb0e000780      mov     ebx,8007000Eh
  176 00007ffc`f5ecde39 e9b269f4ff      jmp     combase!InitMainThreadWnd+0x80 (00007ffc`f5e147f0)  Branch
```

The API function definition is:

```
HWND CreateWindowExW(
  [in]           DWORD     dwExStyle,
  [in, optional] LPCWSTR   lpClassName,
  [in, optional] LPCWSTR   lpWindowName,
  [in]           DWORD     dwStyle,
  [in]           int       X,
  [in]           int       Y,
  [in]           int       nWidth,
  [in]           int       nHeight,
  [in, optional] HWND      hWndParent,
  [in, optional] HMENU     hMenu,
  [in, optional] HINSTANCE hInstance,
  [in, optional] LPVOID    lpParam
);
```

We see the first 4 parameters are passed via ECX, RDX, R8, and R9D according to their size, 32-bit or 64-bit:

```
  148 00007ffc`f5e1477d 4c8d05b45e1d00  lea     r8,[combase!`string' (00007ffc`f5fea638)]
  148 00007ffc`f5e1478b 41b900000088    mov     r9d,88000000h
  148 00007ffc`f5e14791 488b15c0232400  mov     rdx,qword ptr [combase!gOleWindowClass (00007ffc`f6056b58)]
  148 00007ffc`f5e14798 33c9            xor     ecx,ecx
```

R8 contains the 64-bit pointer to a window name (UNICODE):

```
0:000> du 00007ffc`f5fea638
00007ffc`f5fea638  "OleMainThreadWndName"
```

The next 4 parameters are passed via direct stack storage (a default rectangle value):

```
  148 00007ffc`f5e147b0 89442438        mov     dword ptr [rsp+38h],eax
  148 00007ffc`f5e147b4 89442430        mov     dword ptr [rsp+30h],eax
  148 00007ffc`f5e147b8 89442428        mov     dword ptr [rsp+28h],eax
  148 00007ffc`f5e147bc 89442420        mov     dword ptr [rsp+20h],eax
```

It is interesting to see that the last four parameters are passed via direct stack storage as well but using R11 as saved RSP value during function prologue (here, both prologues overlap in purpose):

```
  114 00007ffc`f5e14770 4c8bdc          mov     r11,rsp
  114 00007ffc`f5e14773 53              push    rbx
  114 00007ffc`f5e14774 4883ec60        sub     rsp,60h
  148 00007ffc`f5e14778 498363f000      and     qword ptr [r11-10h],0
  148 00007ffc`f5e1479a 498943e8        mov     qword ptr [r11-18h],rax
  148 00007ffc`f5e147a3 498363e000      and     qword ptr [r11-20h],0
  148 00007ffc`f5e147a8 49c743d8fdffffff mov     qword ptr [r11-28h],0FFFFFFFFFFFFFFFDh
```

Before the *InitMainThreadWnd* function prologue is executed, RSP points to the save return address. So R11 points to that address as well. RSP is then decremented by 8 via the next push instruction. Then 60h bytes (0x60) are reserved, and RSP now points down 0x68 bytes down its original value:

```
R11:                return address
R11-8h  (RSP+60h):  saved RBX
R11-10h (RSP+58h):
R11-18h (RSP+50h):
R11-20h (RSP+48h):
R11-28h (RSP+40h):
RSP+38h:
RSP+30h:
RSP+28h:
RSP+20h:
```

So, by this illustration above, we see the continuity of stack-based parameters despite different CPU registers and offsets used. We also see that all parameters are passed left-to-right.

6. Second, disassemble the *CWnd::CreateEx* function from the *MFC42u* module and find the *CreateWindowExW* API call:

```
0:000> uf MFC42u!CWnd::CreateEx
mfc42u!CWnd::CreateEx:
00007ffc`acc1ca50 488bc4          mov     rax,rsp
00007ffc`acc1ca53 48895808        mov     qword ptr [rax+8],rbx
00007ffc`acc1ca57 48897010        mov     qword ptr [rax+10h],rsi
00007ffc`acc1ca5b 48897818        mov     qword ptr [rax+18h],rdi
00007ffc`acc1ca5f 55              push    rbp
00007ffc`acc1ca60 488d68e1        lea     rbp,[rax-1Fh]
00007ffc`acc1ca64 4881ecb0000000  sub     rsp,0B0h
00007ffc`acc1ca6b 8b4547          mov     eax,dword ptr [rbp+47h]
00007ffc`acc1ca6e 488bd9          mov     rbx,rcx
00007ffc`acc1ca71 8945f7          mov     dword ptr [rbp-9],eax
00007ffc`acc1ca74 8b454f          mov     eax,dword ptr [rbp+4Fh]
00007ffc`acc1ca77 8945f3          mov     dword ptr [rbp-0Dh],eax
00007ffc`acc1ca7a 8b4557          mov     eax,dword ptr [rbp+57h]
00007ffc`acc1ca7d 8945ef          mov     dword ptr [rbp-11h],eax
00007ffc`acc1ca80 8b455f          mov     eax,dword ptr [rbp+5Fh]
00007ffc`acc1ca83 8945eb          mov     dword ptr [rbp-15h],eax
00007ffc`acc1ca86 8b4567          mov     eax,dword ptr [rbp+67h]
00007ffc`acc1ca89 8945e7          mov     dword ptr [rbp-19h],eax
00007ffc`acc1ca8c 488b456f        mov     rax,qword ptr [rbp+6Fh]
00007ffc`acc1ca90 488945df        mov     qword ptr [rbp-21h],rax
00007ffc`acc1ca94 488b4577        mov     rax,qword ptr [rbp+77h]
00007ffc`acc1ca98 488945d7        mov     qword ptr [rbp-29h],rax
00007ffc`acc1ca9c 89550f          mov     dword ptr [rbp+0Fh],edx
00007ffc`acc1ca9f 4c894507        mov     qword ptr [rbp+7],r8
00007ffc`acc1caa3 4c894dff        mov     qword ptr [rbp-1],r9
00007ffc`acc1caa7 e8c4ebffff      call    mfc42u!AfxGetModuleState (00007ffc`acc1b670)
00007ffc`acc1caac 488b4810        mov     rcx,qword ptr [rax+10h]
00007ffc`acc1cab0 488b457f        mov     rax,qword ptr [rbp+7Fh]
00007ffc`acc1cab4 488945c7        mov     qword ptr [rbp-39h],rax
00007ffc`acc1cab8 488b03          mov     rax,qword ptr [rbx]
00007ffc`acc1cabb 48894dcf        mov     qword ptr [rbp-31h],rcx
00007ffc`acc1cabf 49ba709bdb303627698b mov r10,8B69273630DB9B70h
00007ffc`acc1cac9 488b80c8000000  mov     rax,qword ptr [rax+0C8h]
00007ffc`acc1cad0 488d55c7        lea     rdx,[rbp-39h]
00007ffc`acc1cad4 488bcb          mov     rcx,rbx
00007ffc`acc1cad7 ff159b071000    call    qword ptr [mfc42u!_guard_xfg_dispatch_icall_fptr (00007ffc`acd1d278)]
00007ffc`acc1cadd 33ff            xor     edi,edi
00007ffc`acc1cadf 85c0            test    eax,eax
00007ffc`acc1cae1 0f843bc10100    je      mfc42u!CWnd::CreateEx+0x1c1d2 (00007ffc`acc38c22)  Branch

mfc42u!CWnd::CreateEx+0x97:
00007ffc`acc1cae7 488bcb          mov     rcx,rbx
00007ffc`acc1caea e891a5ffff      call    mfc42u!AfxHookWindowCreate (00007ffc`acc17080)
00007ffc`acc1caef 488b45c7        mov     rax,qword ptr [rbp-39h]
00007ffc`acc1caf3 448b4df7        mov     r9d,dword ptr [rbp-9]
00007ffc`acc1caf7 4c8b45ff        mov     r8,qword ptr [rbp-1]
00007ffc`acc1cafb 488b5507        mov     rdx,qword ptr [rbp+7]
00007ffc`acc1caff 8b4d0f          mov     ecx,dword ptr [rbp+0Fh]
00007ffc`acc1cb02 4889442458      mov     qword ptr [rsp+58h],rax
00007ffc`acc1cb07 488b45cf        mov     rax,qword ptr [rbp-31h]
00007ffc`acc1cb0b 4889442450      mov     qword ptr [rsp+50h],rax
00007ffc`acc1cb10 488b45d7        mov     rax,qword ptr [rbp-29h]
00007ffc`acc1cb14 4889442448      mov     qword ptr [rsp+48h],rax
00007ffc`acc1cb19 488b45df        mov     rax,qword ptr [rbp-21h]
00007ffc`acc1cb1d 4889442440      mov     qword ptr [rsp+40h],rax
00007ffc`acc1cb22 8b45e7          mov     eax,dword ptr [rbp-19h]
00007ffc`acc1cb25 89442438        mov     dword ptr [rsp+38h],eax
00007ffc`acc1cb29 8b45eb          mov     eax,dword ptr [rbp-15h]
00007ffc`acc1cb2c 89442430        mov     dword ptr [rsp+30h],eax
00007ffc`acc1cb30 8b45ef          mov     eax,dword ptr [rbp-11h]
00007ffc`acc1cb33 89442428        mov     dword ptr [rsp+28h],eax
00007ffc`acc1cb37 8b45f3          mov     eax,dword ptr [rbp-0Dh]
00007ffc`acc1cb3a 89442420        mov     dword ptr [rsp+20h],eax
00007ffc`acc1cb3e 48ff158bfa0f00  call    qword ptr [mfc42u!_imp_CreateWindowExW (00007ffc`acd1c5d0)]
00007ffc`acc1cb45 0f1f440000      nop     dword ptr [rax+rax]
```

```
00007ffc`acc1cb4a 488bf0          mov     rsi,rax
00007ffc`acc1cb4d e8eea5ffff      call    mfc42u!AfxUnhookWindowCreate (00007ffc`acc17140)
00007ffc`acc1cb52 85c0            test    eax,eax
00007ffc`acc1cb54 0f84ecc00100    je      mfc42u!CWnd::CreateEx+0x1c1f6 (00007ffc`acc38c46)  Branch

mfc42u!CWnd::CreateEx+0x10a:
00007ffc`acc1cb5a 4885f6          test    rsi,rsi
00007ffc`acc1cb5d 400f95c7        setne   dil
00007ffc`acc1cb61 8bc7            mov     eax,edi

mfc42u!CWnd::CreateEx+0x113:
00007ffc`acc1cb63 4c8d9c24b0000000 lea     r11,[rsp+0B0h]
00007ffc`acc1cb6b 498b5b10        mov     rbx,qword ptr [r11+10h]
00007ffc`acc1cb6f 498b7318        mov     rsi,qword ptr [r11+18h]
00007ffc`acc1cb73 498b7b20        mov     rdi,qword ptr [r11+20h]
00007ffc`acc1cb77 498be3          mov     rsp,r11
00007ffc`acc1cb7a 5d              pop     rbp
00007ffc`acc1cb7b c3              ret

mfc42u!CWnd::CreateEx+0x1c1d2:
00007ffc`acc38c22 488b03          mov     rax,qword ptr [rbx]
00007ffc`acc38c25 49ba7011d03a0b36b89a mov r10,9AB8360B3AD01170h
00007ffc`acc38c2f 488b8058010000  mov     rax,qword ptr [rax+158h]
00007ffc`acc38c36 488bcb          mov     rcx,rbx
00007ffc`acc38c39 ff1539460e00    call    qword ptr [mfc42u!_guard_xfg_dispatch_icall_fptr (00007ffc`acd1d278)]
00007ffc`acc38c3f 33c0            xor     eax,eax
00007ffc`acc38c41 e91d3ffeff      jmp     mfc42u!CWnd::CreateEx+0x113 (00007ffc`acc1cb63)  Branch

mfc42u!CWnd::CreateEx+0x1c1f6:
00007ffc`acc38c46 488b0b          mov     rcx,qword ptr [rbx]
00007ffc`acc38c49 488b8158010000  mov     rax,qword ptr [rcx+158h]
00007ffc`acc38c50 49ba7011d03a0b36b89a mov r10,9AB8360B3AD01170h
00007ffc`acc38c5a 488bcb          mov     rcx,rbx
00007ffc`acc38c5d ff1515460e00    call    qword ptr [mfc42u!_guard_xfg_dispatch_icall_fptr (00007ffc`acd1d278)]
00007ffc`acc38c63 90              nop
00007ffc`acc38c64 e9f13efeff      jmp     mfc42u!CWnd::CreateEx+0x10a (00007ffc`acc1cb5a)  Branch
```

We see that 8 parameters are passed using RSP pointer and consecutive offsets.

7. Let's look at an indirect call through IAT:

```
0:000> dps 00007ffc`acd1c5d0 L1
00007ffc`acd1c5d0  00007ffc`f5848030 user32!CreateWindowExW

0:000> !dh mfc42u

File Type: DLL
FILE HEADER VALUES
    8664 machine (X64)
       7 number of sections
F91A937D time date stamp Fri Jun  9 04:54:37 2102

       0 file pointer to symbol table
       0 number of symbols
      F0 size of optional header
    2022 characteristics
            Executable
            App can handle >2gb addresses
            DLL

OPTIONAL HEADER VALUES
     20B magic #
   14.28 linker version
   F2000 size of code
   81000 size of initialized data
       0 size of uninitialized data
   21730 address of entry point
```

```
    1000 base of code
         ----- new -----
00007ffcacc10000 image base
    1000 section alignment
    1000 file alignment
       3 subsystem (Windows CUI)
   10.00 operating system version
   10.00 image version
   10.00 subsystem version
  174000 size of image
    1000 size of headers
  17BE15 checksum
0000000000040000 size of stack reserve
0000000000001000 size of stack commit
0000000000100000 size of heap reserve
0000000000001000 size of heap commit
    4160  DLL characteristics
            High entropy VA supported
            Dynamic base
            NX compatible
            Guard
  139980 [    6CC0] address [size] of Export Directory
  140640 [     21C] address [size] of Import Directory
  163000 [    A4B8] address [size] of Resource Directory
  151000 [   100A4] address [size] of Exception Directory
       0 [       0] address [size] of Security Directory
  16E000 [    5734] address [size] of Base Relocation Directory
  116B60 [      70] address [size] of Debug Directory
       0 [       0] address [size] of Description Directory
       0 [       0] address [size] of Special Directory
   FD4A0 [      28] address [size] of Thread Storage Directory
   FBC00 [     138] address [size] of Load Configuration Directory
       0 [       0] address [size] of Bound Import Directory
  10BD18 [    1548] address [size] of Import Address Table Directory
  13838C [     180] address [size] of Delay Import Directory
       0 [       0] address [size] of COR20 Header Directory
       0 [       0] address [size] of Reserved Directory


SECTION HEADER #1
   .text name
   F124E virtual size
    1000 virtual address
   F2000 size of raw data
    1000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
60000020 flags
         Code
         (no align specified)
         Execute Read

SECTION HEADER #2
  .rdata name
   51894 virtual size
   F3000 virtual address
   52000 size of raw data
   F3000 file pointer to raw data
```

```
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
40000040 flags
         Initialized Data
         (no align specified)
         Read Only


Debug Directories(4)
       Type       Size     Address  Pointer
       cv           23      126810   126810    Format: RSDS, guid, 1, mfc42u.pdb
       (   13)     5e4      126834   126834
       (   16)      24      126e18   126e18
       dllchar       4      126e3c   126e3c

00000001 extended DLL characteristics
          CET compatible


SECTION HEADER #3
   .data name
    B19C virtual size
  145000 virtual address
    5000 size of raw data
  145000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
C0000040 flags
         Initialized Data
         (no align specified)
         Read Write

SECTION HEADER #4
  .pdata name
   100A4 virtual size
  151000 virtual address
   11000 size of raw data
  14A000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
40000040 flags
         Initialized Data
         (no align specified)
         Read Only

SECTION HEADER #5
  .didat name
     548 virtual size
  162000 virtual address
    1000 size of raw data
  15B000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
```

```
       0 number of line numbers
C0000040 flags
         Initialized Data
         (no align specified)
         Read Write

SECTION HEADER #6
   .rsrc name
    A4B8 virtual size
  163000 virtual address
    B000 size of raw data
  15C000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
40000040 flags
         Initialized Data
         (no align specified)
         Read Only

SECTION HEADER #7
  .reloc name
    5734 virtual size
  16E000 virtual address
    6000 size of raw data
  167000 file pointer to raw data
       0 file pointer to relocation table
       0 file pointer to line numbers
       0 number of relocations
       0 number of line numbers
42000040 flags
         Initialized Data
         Discardable
         (no align specified)
         Read Only
```

We see that the *mfc42u!_imp_CreateWindowExW* address is inside the IAT memory region of length 0x1548 bytes:

```
0:000> ? 00007ffcacc10000 + 10BD18
Evaluate expression: 140723207912728 = 00007ffc`acd1bd18

0:000> * mfc42u!_imp_CreateWindowExW (00007ffc`acd1c5d0)

0:000> ? 00007ffcacc10000 + 10BD18 + 1548
Evaluate expression: 140723207918176 = 00007ffc`acd1d260
```

8. We close logging before exiting WinDbg Preview:

```
0:000> .logclose
Closing open log file C:\AWAPI-Dumps\W1.log
```

Note: To avoid possible confusion and glitches, we recommend exiting WinDbg Preview or WinDbg after each exercise.

9. Let's now look at a 32-bit process. Launch the new instance of WinDbg Preview.

10. Open \AWAPI-Dumps\Process\x86\wordpad.DMP

11. We get the dump file loaded:

```
Microsoft (R) Windows Debugger Version 10.0.25200.1003 X86
Copyright (c) Microsoft Corporation. All rights reserved.


Loading Dump File [C:\AWAPI-Dumps\Process\x86\wordpad.DMP]
User Mini Dump File with Full Memory: Only application data is available


************* Path validation summary **************
Response                         Time (ms)     Location
Deferred                                       srv*
Symbol search path is: srv*
Executable search path is:
Windows 10 Version 22000 MP (2 procs) Free x86 compatible
Product: WinNt, suite: SingleUserTS Personal
Edition build lab: 22000.1.amd64fre.co_release.210604-1628
Machine Name:
Debug session time: Fri Nov  4 21:05:05.000 2022 (UTC + 0:00)
System Uptime: 0 days 0:39:06.687
Process Uptime: 20 days 0:46:35.000
................................................................
.....
Loading unloaded module list
.
For analysis of this file, run !analyze -v
eax=00000000 ebx=010593c8 ecx=00000000 edx=00000000 esi=010593fc edi=010593fc
eip=758c10cc esp=0013f870 ebp=0013f8a8 iopl=0         nv up ei pl nz ac po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000212
win32u!NtUserGetMessage+0xc:
758c10cc c21000          ret     10h
```

12. Open the same log file using **.logappend**:

```
0:000> .logappend C:\AWAPI-Dumps\W1.log
Opened log file 'C:\AWAPI-Dumps\W1.log'
```

13. Let's look at the *InitMainThreadWnd* function from the *combase* module and find the *CreateWindowExW* API call:

```
0:000> uf combase!InitMainThreadWnd
0:000> uf combase!InitMainThreadWnd
combase!InitMainThreadWnd [onecore\com\combase\object\mainthrd.cxx @ 114]:
  114 75c032e3 8bff            mov     edi,edi
  114 75c032e5 51              push    ecx
  114 75c032e6 56              push    esi
  115 75c032e7 33f6            xor     esi,esi
  115 75c032e9 57              push    edi
  115 75c032ea 46              inc     esi
  125 75c032eb e8235ef8ff      call    combase!IsWOWThread (75b89113)
  125 75c032f0 85c0            test    eax,eax
  125 75c032f2 7546            jne     combase!InitMainThreadWnd+0x57 (75c0333a)  Branch

combase!InitMainThreadWnd+0x11 [onecore\com\combase\object\mainthrd.cxx @ 148]:
  148 75c032f4 33ff            xor     edi,edi
  148 75c032f6 b800000080      mov     eax,80000000h
  148 75c032fb 57              push    edi
  148 75c032fc ff35e8ccd275    push    dword ptr [combase!g_hinst (75d2cce8)]
  148 75c03302 57              push    edi
  148 75c03303 6afd            push    0FFFFFFFDh
```

```
  148 75c03305 50              push    eax
  148 75c03306 50              push    eax
  148 75c03307 50              push    eax
  148 75c03308 50              push    eax
  148 75c03309 6800000088      push    88000000h
  148 75c0330e 6870f0b075      push    offset combase!`string' (75b0f070)
  148 75c03313 ff353cd3d275    push    dword ptr [combase!gOleWindowClass (75d2d33c)]
  148 75c03319 57              push    edi
  148 75c0331a ff15e842d375    call    dword ptr [combase!_imp__CreateWindowExW (75d342e8)]
  148 75c03320 a340d3d275      mov     dword ptr [combase!ghwndOleMainThread (75d2d340)],eax
  163 75c03325 85c0            test    eax,eax
  163 75c03327 0f848aa30800    je      combase!InitMainThreadWnd+0x8a3d4 (75c8d6b7)  Branch

combase!InitMainThreadWnd+0x4a [onecore\com\combase\object\mainthrd.cxx @ 180]:
  180 75c0332d ff15ecf3d275    call    dword ptr [combase!_imp__GetCurrentThreadId (75d2f3ec)]
  180 75c03333 a334d3d275      mov     dword ptr [combase!gdwMainThreadId (75d2d334)],eax
  181 75c03338 8bf7            mov     esi,edi

combase!InitMainThreadWnd+0x57 [onecore\com\combase\object\mainthrd.cxx @ 187]:
  187 75c0333a 5f              pop     edi
  187 75c0333b 8bc6            mov     eax,esi
  187 75c0333d 5e              pop     esi
  187 75c0333e 59              pop     ecx
  187 75c0333f c3              ret

combase!InitMainThreadWnd+0x8a3d4 [onecore\com\combase\object\mainthrd.cxx @ 168]:
  168 75c8d6b7 ff1538f2d275    call    dword ptr [combase!_imp__GetLastError (75d2f238)]
  168 75c8d6bd 8bf0            mov     esi,eax
  168 75c8d6bf 85f6            test    esi,esi
  168 75c8d6c1 7e09            jle     combase!InitMainThreadWnd+0x8a3e9 (75c8d6cc)  Branch

combase!InitMainThreadWnd+0x8a3e0 [onecore\com\combase\object\mainthrd.cxx @ 168]:
  168 75c8d6c3 0fb7f6          movzx   esi,si
  168 75c8d6c6 81ce00000780    or      esi,80070000h

combase!InitMainThreadWnd+0x8a3e9 [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 75c8d6cc 393d70c2d275    cmp     dword ptr [combase!_Tlgg_hCombaseTraceLoggingProviderProv (75d2c270)],edi
  169 75c8d6d2 7713            ja      combase!InitMainThreadWnd+0x8a404 (75c8d6e7)  Branch

combase!InitMainThreadWnd+0x8a3f1 [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 75c8d6d4 393d48ced275    cmp     dword ptr [combase!gfEnableTracing (75d2ce48)],edi
  169 75c8d6da 742a            je      combase!InitMainThreadWnd+0x8a423 (75c8d706)  Branch

combase!InitMainThreadWnd+0x8a3f9 [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 75c8d6dc 33c9            xor     ecx,ecx
  169 75c8d6de e86a4df2ff      call    combase!IsWppLevelEnabled (75bb244d)
  169 75c8d6e3 84c0            test    al,al
  169 75c8d6e5 741f            je      combase!InitMainThreadWnd+0x8a423 (75c8d706)  Branch

combase!InitMainThreadWnd+0x8a404 [onecore\com\combase\object\mainthrd.cxx @ 169]:
  169 75c8d6e7 56              push    esi
  169 75c8d6e8 682cabb075      push    offset combase!`string' (75b0ab2c)
  169 75c8d6ed 57              push    edi
  169 75c8d6ee 68a9000000      push    0A9h
  169 75c8d6f3 6854eeb375      push    offset combase!`string' (75b3ee54)
  169 75c8d6f8 687ce6b175      push    offset combase!`string' (75b1e67c)
  169 75c8d6fd 56              push    esi
  169 75c8d6fe e8662dedff      call    combase!ComTraceMessage (75b60469)
  169 75c8d703 83c41c          add     esp,1Ch

combase!InitMainThreadWnd+0x8a423 [onecore\com\combase\object\mainthrd.cxx @ 170]:
  170 75c8d706 85f6            test    esi,esi
  170 75c8d708 0f882c5cf7ff    js      combase!InitMainThreadWnd+0x57 (75c0333a)  Branch

combase!InitMainThreadWnd+0x8a42b [onecore\com\combase\object\mainthrd.cxx @ 172]:
  172 75c8d70e be0e000780      mov     esi,8007000Eh
  176 75c8d713 e9225cf7ff      jmp     combase!InitMainThreadWnd+0x57 (75c0333a)  Branch
```

To remind you, the API function definition is:

```
HWND CreateWindowExW(
  [in]           DWORD     dwExStyle,
  [in, optional] LPCWSTR   lpClassName,
  [in, optional] LPCWSTR   lpWindowName,
  [in]           DWORD     dwStyle,
  [in]           int       X,
  [in]           int       Y,
  [in]           int       nWidth,
  [in]           int       nHeight,
  [in, optional] HWND      hWndParent,
  [in, optional] HMENU     hMenu,
  [in, optional] HINSTANCE hInstance,
  [in, optional] LPVOID    lpParam
);
```

We see that all parameters are pushed right-to-left with the third parameter from the left containing the 64-bit pointer to a window name (UNICODE):

```
0:000> du 75b0f070
75b0f070  "OleMainThreadWndName"
```

14. We close logging before exiting WinDbg Preview:

```
0:000> .logclose
Closing open log file C:\AWAPI-Dumps\W1.log
```