Image File Execution Options just inserts the debugger in front of the command line

devblogs.microsoft.com/oldnewthing/20070702-00

July 2, 2007

Raymond Chen

If you use the Image File Execution Options registry key to force a program to run under the debugger, all the kernel does is insert the debugger in front of the command line. In other words, the `CreateProcess` function figure out what program is about to be run and checks the Image File Execution Options. If it finds a debugger, then the debugger is prepended to the command line and then `CreateProcess` resumes as if that were the command line you had passed originally.

In particular, it doesn't do anything with the other parameters to the `CreateProcess` function. If you passed special parameters via the `STARTUPINFO` structure, those parameters get passed to the debugger. And the `PROCESS_INFO` that is returned by the `CreateProcess` function describes the debugger, not the process being debugged.

Specifically, if you specified the `STARTF_USESHOWWINDOW` flag and passed, say, `SW_HIDE`, as the `wShowWindow`, then *the debugger will be hidden*. This bites me every so often when I am called upon to debug a program that happens to be launched as hidden. I'll slap the debugger underneath it with Image File Execution Options, run through the scenario, and then... nothing.

And then eventually I realize, "Oh, right, the debugger is hidden."

To unstick myself, I fire up a program like Spy to get the window handle of the hidden debugger and fire up a scratch copy of Notepad so I can make it do my bidding and show the window.

```
ntsd -Ggx notepad
<F12>
Break instruction exception - code 80000003 (first chance)
eax=7ffdf000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c901230 esp=00a1ffcc ebp=00a1fff4 iopl=0         nv up ei pl zr na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c901230 cc               int     3
0:001> r esp=esp-4
0:001> ed esp 1
0:001> r esp=esp-4
0:001> ed esp 0x00010164
0:001> r esp=esp-4
0:001> ed esp eip
0:001> r eip=user32!showwindow
0:001> g
0:001> q
```

The first two commands push the value `SW_SHOWNORMAL` (numerical value 1) onto the stack. Then goes the window handle. And then the return address. Move the instruction pointer to `user32!ShowWindow` and we've simulated the function call `ShowWindow(0x00010164, SW_SHOWNORMAL);`. Once I let execution resume, *boom* the debugger window appears and I can continue my work.

Raymond Chen

Follow