# Psychic debugging: IP on heap

devblogs.microsoft.com/oldnewthing/20071114-00

November 14, 2007

Raymond Chen

Somebody asked the shell team to look at this crash in a context menu shell extension.

```
IP_ON_HEAP:  003996d0
ChildEBP RetAddr
00b2e1d8 68f79ca6 0x3996d0
00b2e1f4 7713a7bd ATL::CWindowImplBaseT<
                          ATL::CWindow,ATL::CWinTraits<2147483648,0> >
                     ::StartWindowProc+0x43
00b2e220 77134be0 USER32!InternalCallWinProc+0x23
00b2e298 7713a967 USER32!UserCallWinProcCheckWow+0xe0
...
eax=68f79c63 ebx=00000000 ecx=00cade10 edx=7770df14 esi=002796d0 edi=000603cc
eip=002796d0 esp=00cade4c ebp=00cade90 iopl=0         nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010206
002796d0 c744240444bafb68 mov     dword ptr [esp+4],68fbba44
```

You should be able to determine the cause instantly.

I replied,

> This shell extension is using a non-DEP-aware version of ATL. They need to upgrade to ATL 8 or disable DEP.

This was totally obvious to me, but the person who asked the question met it with stunned amazement. I guess the person forgot that older versions of ATL are notorious DEP violators. You see a DEP violation, you see that it's coming from ATL, and bingo, you have your answer. When DEP was first introduced, the base team sent out mail to the entire Windows division saying, "Okay, folks, we're turning it on. You're going to see a lot of application compatibility problems, especially this ATL one."

Psychic powers sometimes just means having a good memory.

Even if you forgot that information, it's still totally obvious once you look at the scenario and understand what it's trying to do.

The fault is `IP_ON_HEAP` which is precisely what DEP protects against. The next question is why IP ended up on the heap. Was it a mistake or intentional?

Look at the circumstances surrounding the faulting instruction again. The faulting instruction is the window procedure for a window, and the action is storing a constant into the stack. The symbols of the caller tell us that it's some code in ATL, and you can even go look up the source code yourself:

```
template <class TBase, class TWinTraits>
LRESULT CALLBACK CWindowImplBaseT< TBase, TWinTraits >
  ::StartWindowProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {
    CWindowImplBaseT< TBase, TWinTraits >* pThis =
              (CWindowImplBaseT< TBase, TWinTraits >*)
                  _AtlWinModule.ExtractCreateWndData();
    pThis->m_hWnd = hWnd;
    pThis->m_thunk.Init(pThis->GetWindowProc(), pThis);
    WNDPROC pProc = pThis->m_thunk.GetWNDPROC();
    ::SetWindowLongPtr(hWnd, GWLP_WNDPROC, (LONG_PTR)pProc);
    return pProc(hWnd, uMsg, wParam, lParam);
}
```

Is `pProc` corrupted and we're jumping to a random address on the heap? Or was this intentional?

ATL is clearly generating code on the fly (the window procedure thunk), and it is in execution of the thunk that we encounter the DEP exception.

Now, you didn't need to have the ATL source code to realize that this is what's going on. It is a very common pattern in framework libraries to put a C++ wrapper around window procedures. Since C++ functions have a hidden `this` parameter, the wrappers need to sneak that parameter in somehow, and one common technique is to generate some code on the fly that sets up the hidden `this` parameter before calling the C++ function. The value at `[esp+4]` is the window handle, something that can be recovered from the `this` pointer, so it's a handly thing to replace with `this` before jumping to the real C++ function.

The address being stored as the `this` parameter is `68fbba44`, which is inside the DLL in question. (You can tell this because the return address, which points to the ATL thunk code, is at `68f79ca6` which is in the same neighborhood as the mystery pointer.) Therefore, this is almost certainly an ATL thunk for a static C++ object.

In other words, this is extremely unlikely be a jump to a random address. The code at the address looks too good. It's probably jumping there intentionally, and the fact that it's coming from a window procedure thunk confirms it.

But our tale is not over yet. The plot thickens. We'll continue next time.

Raymond Chen

## Follow