Debugging session: Which of the many things happening in this single line of code is the one that crashed?

devblogs.microsoft.com/oldnewthing/20160526-00

May 26, 2016

Raymond Chen

A crash report came in, and the offending line of code was the following:

```
void CDeloreanSettings::UpdateFluxModulation(bool sendNotification)
{
    ComPtr<IFluxModulator> spModulator;
    // Crash on the next line
    if (SUCCEEDED(m_spFluxCapacitor->GetFluxModulator(&spModulator)))
    {
        ...
    }
}
```

Someone made the initial diagnosis that

> The call is to `ReleaseAndGetAddressOf()` on a `ComPtr` object which is declared right above (which should be initialized to `nullptr`). Am I missing something?

Let's look at the disassembly. First, with no annotations. See if you can figure it out yourself.

```
CDeloreanSettings::UpdateFluxModulation:
mov     qword ptr [rsp+10h],rbx
mov     qword ptr [rsp+18h],rsi
mov     qword ptr [rsp+20h],rdi
push    rbp
push    r14
push    r15
mov     rbp,rsp
sub     rsp,50h
mov     rax,qword ptr [__security_cookie]
xor     rax,rsp
mov     qword ptr [rbp-8],rax
mov     rdi,qword ptr [rcx+18h]
mov     r14,rcx
lea     rcx,[rbp-10h]
xor     esi,esi
mov     r15b,dl
and     qword ptr [rbp-10h],rsi
call    Microsoft::WRL::ComPtr<IUnrelatedInterface>::InternalRelease
mov     rax,qword ptr [rdi] << crash here
mov     rbx,qword ptr [rax+38h]
mov     rcx,rbx
call    qword ptr [__guard_check_icall_fptr]
lea     rdx,[rbp-10h]
mov     rcx,rdi
call    rbx
```

Okay, here’s the version with my annotations:

```
CDeloreanSettings::UpdateFluxModulation:
; Prologue: Save nonvolatile registers and build the stack frame.
mov     qword ptr [rsp+10h],rbx
mov     qword ptr [rsp+18h],rsi
mov     qword ptr [rsp+20h],rdi
push    rbp
push    r14
push    r15
mov     rbp,rsp
sub     rsp,50h
mov     rax,qword ptr [__security_cookie]
xor     rax,rsp
mov     qword ptr [rbp-8],rax

mov     rdi,qword ptr [rcx+18h] ; rdi = m_spFluxCapacitor
mov     r14,rcx                 ; save "this"
lea     rcx,[rbp-10h]           ; prepare spModulator.ReleaseAndGetAddressOf
xor     esi,esi
mov     r15b,dl                 ; save "sendNotification"
and     qword ptr [rbp-10h],rsi ; construct spModulator
; ReleaseAndGetAddressOf was inlined. Here's the Release part:
call    Microsoft::WRL::ComPtr<IUnrelatedInterface>::InternalRelease

; prepare m_spFluxCapacitor->...
; Crash here loading vtable from m_spFluxCapacitor
mov     rax,qword ptr [rdi] << crash here
mov     rbx,qword ptr [rax+38h] ; load address of GetFluxModulator
mov     rcx,rbx                 ; parameter to CFG check
call    qword ptr [__guard_check_icall_fptr] ; check the function pointer

; Here's the GetAddressOf part of ReleaseAndGetAddressOf:
lea     rdx,[rbp-10h] ; spModulator.GetAddressOf
mov     rcx,rdi                 ; "this" for GetFluxModulator
call    rbx                     ; _spFluxCapacitor->GetFluxModulator()
```

The compiler inlined `ReleaseAndGetAddressOf`, and it interleaved various unrelated operations. In the second block of code, you can see it interleave the construction of the `ComPtr` with the call to `InternalRelease`. In the third block, you can see it peform the control flow guard test before performing the `GetAddresssOf`.

The conclusion, therefore, is not that the crash occurred in the `ReleaseAndGetAddressOf` The `ReleaseAndGetAddressOf` just finished releasing and is waiting for its turn to do the `GetAddresssOf`. Rather, the crash occurred because `m_spFluxCapacitor` is null, and we crashes trying to read the vtable from a null pointer.

Further investigation of the issue revealed that `UpdateFluxModulation` is called from an event handler that was registered to be called whenever the modulation changed. Inspection of memory showed that the event registration token was zero, indicating that the event has already been unregistered. The issue is that there was a modulation change in flight when the event handler was unregistered, so the `CDeloreanSettings` received its change notification after it had unregistered. The fix is to have the handler check whether it still has a `m_spFluxCapacitor`, and if not, then ignore the notification, on the assumption that it was a stray notification that was late to arrive.

Raymond Chen

Follow