The case of the fail-fast crashes coming from the power management system

devblogs.microsoft.com/oldnewthing/20240913-00

September 13, 2024

A customer reported that they were seeing around four thousand crashes a day from an internal function `RtlpHandleInvalidUserCallTarget`. Here's one of their crash dumps:

```
Child-SP          RetAddr               Call Site
0000009b`c5bfefc8 00007ffd`e32ed7ad     ntdll!RtlFailFast2
0000009b`c5bfefd0 00007ffd`e327c798     ntdll!RtlpHandleInvalidUserCallTarget+0x5d
0000009b`c5bff000 00007ffd`dfdf2dee     ntdll!LdrpHandleInvalidUserCallTarget+0x38
(Inline Function) --------`--------     powrprof!PowerpNotifyCallbackSafe+0x13
```

The `RtlpHandleInvalidUserCallTarget` function is used by Control Flow Guard when it detects that somebody is trying to call an invalid function pointer. So what is the invalid pointer?

Since debugging is an exercise in optimism, let's hope that the pointer is still in one of the registers.

```
0:103> r
rax=0000000000000000 rbx=00007ffd5fe8be40 rcx=000000000000000a
rdx=00007ffd5fe8be40 rsi=0000000000000004 rdi=0000000000000000
rip=00007ffde3292350 rsp=0000009bc5bfefc8 rbp=0000000000000000
 r8=0000000000000000  r9=0000000000000003 r10=0000000000000001
r11=0000000000000000 r12=0000000000000000 r13=0000000000000000
r14=0000009bc5bffa98 r15=000000007ffe03b0
ntdll!RtlFailFast2:
0:103>
```

There are only two things that look like possible function pointers,[1] and they are both equal, so let's see if we're lucky.

```
0:103> u @rbx L1
<Unloaded_ContosoVirtualCamera.dll>+0x7be40:
00007ffd`5fe8be40 ??              ???
```

Bingo. Got it in one.

It kind of makes sense that we'd find the function pointer in the `rdx` register, since that holds the second function parameter. (The first function parameter is `rcx`, which holds the fail-fast code `0x0000000A`:

```
#define FAST_FAIL_GUARD_ICALL_CHECK_FAILURE         10
```

which tells us that we have a CFG failure. So it's not too surprising that the second parameter is the pointer that failed validation.)

If we wanted to be more methodical about it, we could look where the function pointer got saved. Let's look at the code in `RtlpHandleInvalidUserCallTarget` up to the point where it called `RtlFailFast2` and see if we can follow where the function pointer went. The goal is to find a path from the start of the function to the `RtlFailFast2`, so I'll highlight that path and de-emphasize the rest.

```
ntdll!RtlpHandleInvalidUserCallTarget:
    push    rbx
    sub     rsp,20h
    cmp     byte ptr [00007ffd`e33712a2],0
    mov     rbx,rcx ← saved rcx in rbx
    je      00007ffd`e32ed77f
    call    ntdll!RtlpGuardIsSuppressedAddress (00007ffd`e32ed720)
    test    al,al
    je      00007ffd`e32ed77f
    mov     edx,1
    mov     rcx,rbx
    call    ntdll!RtlpGuardGrantSuppressedCallAccess (00007ffd`e32375b8)
00007ffd`e32ed778:
    add     rsp,20h
    pop     rbx
    ret
    int     3
00007ffd`e32ed77f:
    call    ntdll!LdrControlFlowGuardEnforcedWithExportSuppression
(00007ffd`e32234e8)
    test    eax,eax
    je      00007ffd`e32ed7a0
    mov     rcx,rbx
    call    ntdll!RtlGuardIsExportSuppressedAddress (00007ffd`e323765c)
    test    al,al
    je      00007ffd`e32ed7a0
    mov     rcx,rbx
    call    ntdll!RtlpUnsuppressForwardReferencingCallTarget (00007ffd`e32ed7b4)
    test    eax,eax
    jns     00007ffd`e32ed778
00007ffd`e32ed7a0:
    mov     rdx,rbx ← rbx moved to rdx
    mov     ecx,0Ah
    call    ntdll!RtlFailFast2 (00007ffd`e3292350)
```

By following the flow, we see that the inbound `rcx` was saved in `rbx`, and then copied back to `rdx` for the fail-fast. So that’s where the function pointer is, and that also explains why we see the same value in both `rbx` and `rdx`.

The conclusion, therefore, is that the Contoso virtual camera driver registered a power management callback (hard to tell which one, but it’s going to be `PowerRegisterSuspend-ResumeNotification` or something like that), and they forgot to unregister it before their DLL unloaded. And then the power event occurred, and the power management system calls a callback that points to an unloaded DLL.

So the next step here is to reach out to Contoso and let them know about the crashing bug in their virtual camera driver. Meanwhile, the customer can put the buggy versions of the Contoso virtual camera driver on their “do not use” list.

[1] Well, three if you count `rip`, but that's not interesting because that's the current instruction pointer!