Sucking the trap frame out of a kernel mode stack trace

devblogs.microsoft.com/oldnewthing/20081024-00

October 24, 2008

Raymond Chen

If you are placed in the unfortunate position of having to debug a user-mode crash from kernel mode, one of the first things you have to do is get back to the exception on the user-mode side so you can see what happened. We saw earlier how you can get symbols for operating system binaries to help you suck the exception pointers out of a user-mode stack trace; here's a corresponding tip for the kernel-mode side.

Your stack trace will look something like this:

```
ChildEBP RetAddr  Args to Child
8fc86660 818844e3 83811e00 83811d78 83811e30 nt!KiSwapContext+0x26
8fc8669c 8184abd2 83811d78 00000000 83811d78 nt!KiSwapThread+0x3d2
8fc866fc 81a690b1 8fc86740 00000000 00000000 nt!KeWaitForSingleObject+0x414
8fc8681c 81a6a5aa 90a06108 83811d78 8fc86860 nt!DbgkpQueueMessage+0x283
8fc86844 819e3cbd 8fc86860 80000003 00000000 nt!DbgkpSendApiMessage+0x44
8fc86908 8183c542 8fc86cf0 00000001 00000000 nt!DbgkForwardException+0xd0
8fc86cd4 8184e51a 8fc86cf0 00000000 8fc86d44 nt!KiDispatchException+0x2ee
8fc86d3c 8184e4ce 063fedc8 75b9b7df badb0d00 nt!CommonDispatchException+0x4a
8fc86d44 75b9b7df badb0d00 00000000 00000000 nt!KiExceptionExit+0x186
063fedc8 75b963ea 4eedcfb0 4f370fb0 063ff0bc ABC!Control::Character::OnDestroy+0xbc
063ff020 747d3782 4f370fb0 5665cf68 063ff0bc
ABC!Control::Character::MessageHandler+0x476
063ff034 747d3819 063ff0bc 063ff050 747d37f6 DEF!EventGizmo::FireEvent+0xf
063ff040 747d37f6 063ff0bc 0000000c 063ff0a8 DEF!Gizmo::CallStubEvent+0x1a
063ff050 747d3842 4f370fb0 063ff0bc c6db9237 DEF!Callback::CallOnEvent+0x19
063ff0a8 747d6ed0 4f370fb0 063ff0bc 00000001 DEF!Callback::Invoke+0x20
063ff0d0 747d7708 4f370fb0 00000001 4eedcfb0 DEF!Callback::FireDestroy+0x2a
063ff0f0 747d728a 3618af68 4eedcfb0 747d7429
DEF!ObjectManager::DestroyAllChildren+0x34
063ff0fc 747d7429 4a1bff78 4eedcfb0 747d6e9d DEF!ObjectManager::BeginDestroy+0x2e
063ff108 747d6e9d 4eedcfb0 747d7721 4a1bff78 DEF!ObjectManager::Destroy+0x1a
```

The third parameter to `KiDispatchException` is the trap frame. Most people who write about `KiDispatchException` do so in the context of driver debugging, but trap frames are also used during user-mode-to-kernel-mode transitions.

```
0: kd> .trap 8fc86d44
ErrCode = 00000004
eax=00000001 ebx=00000001 ecx=4e62e594 edx=00000000 esi=5665cf68 edi=5630eff8
eip=75b9b7df esp=063fedb4 ebp=063fedc8 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00010246
ABC!Control::Character::OnDestroy+0xbc:
001b:75b9b7df 8b01            mov     eax,dword ptr [ecx] ds:0023:4e62e594=?????
???
```

And there you have it, the original exception.

Raymond Chen

Follow