The case of Explorer calling into an unloaded DLL while trying to run down a reference to it

devblogs.microsoft.com/oldnewthing/20200814-00

August 14, 2020

Raymond Chen

There was a large number of crashes in Explorer that were tracked back to attempting to release a COM object that belonged to a DLL that was no longer in memory.

A typical call stack at the crash looked like this:

```
combase!<lambda_...>::operator()+0x9e
combase!ObjectMethodExceptionHandlingAction<lambda_...>+0x1b
combase!CStdIdentity::ReleaseCtrlUnk+0x68
combase!CStdMarshal::DisconnectWorker_ReleasesLock+0x385
combase!CStdMarshal::DisconnectSwitch_ReleasesLock+0x28
combase!CStdMarshal::DisconnectAndReleaseWorker_ReleasesLock+0x3c
combase!CStdMarshal::DisconnectAndRelease+0x35
combase!COIDTable::ThreadCleanup+0xd5
combase!FinishShutdown::<lambda_...>::operator()+0x5
combase!ObjectMethodExceptionHandlingAction<lambda_...>+0x15
combase!FinishShutdown+0x45
combase!ApartmentUninitialize+0x67
combase!wCoUninitialize+0x11a
combase!CoUninitialize+0xb6
imm32!CtfImmCoUninitialize+0x48
msctf!CicFlsCallback+0x50
ntdll!RtlProcessFlsData+0xf6
ntdll!LdrShutdownThread+0x32
ntdll!RtlExitUserThread+0x4c
KERNELBASE!FreeLibraryAndExitThread+0x34
ucrtbase!common_end_thread+0x84
ucrtbase!_endthreadex+0x7
ucrtbase!thread_start+0x46
kernel32!BaseThreadInitThunk+0x24
ntdll!__RtlUserThreadStart+0x2f
ntdll!_RtlUserThreadStart+0x1b
```

I took a sample of ten crashes with this stack to see if I could find a pattern. The object being released is still alive (the data for it is still present in memory, and it still has a vtable), but the code that the vtable points to has already been unloaded. Fortunately, the system

remembers the DLLs that were most recently unloaded, so we can use that to look up the DLLs that hosted the objects that are being run down.

The ten crashes break down like this:

Number	Culprit
7	Contoso
2	Fabrikam
1	LitWare

The vast majority of the issues are with Contoso, so we'll focus on that one.

An interesting detail is that in four of the Contoso crashes, some version of the Contoso setup program is running.

I got lucky and discovered that Contoso is an open source project, so I was able to make further progress by reading the code and seeing what they were trying to do.

Contoso injects its DLL into Explorer and takes over a bunch of stuff. When Contoso wants to unload from Explorer, it unhooks all the hooks that it installed and unloads itself. It wasn't loaded by COM, so COM is not going to call `DllCanUnloadNow` to see whether the DLL has any active COM objects that would require it to remain loaded in memory.

However, it does produce COM objects, particularly, implementations of `IAccessible` so that its UI objects are available to screen readers and other UI automation clients.

Once I had this foothold, it was relatively easy to reproduce the problem:

- Start Narrator.
- Launch the Contoso UI.
- Uninstall Contoso.
- Perform a developer shutdown of Explorer; `Ctrl`+`Shift`+RightClick, *Exit Explorer*.

Here's what's going on.

Launching the Contoso UI causes Narrator to ask for the `IAccessible` interface so it can navigate the user interface elements.

Uninstalling Contoso causes it to remove its injected DLL, even though there are `IAccessible` objects still outstanding. These are ticking time bombs waiting to be triggered.[1]

Shutting down Explorer causes COM to be shut down for the process, at which time it runs down all the outstanding objects. And that's when it trips over these `IAccessible` objects that are backed by code that is no longer present in the process.

The fix is to create a custom COM context to hold your objects so that you can disconnect them prior to unloading. And the project owners agreed to make a fix to do exactly that.

One of the burdens of Explorer is that it is an attractive target for third-party code to inject itself, despite it being totally unsupported. And when that third-party code crashes, it's Explorer that takes the blame.

One crash caused by a third party code-injector down. A few million more to go.

[1] That explains why the Contoso installer is often running at the time of these crashes. One of the things that the Contoso installer does is uninstall the previous version.

Raymond Chen

Follow