Could there be any problems with calling GetModuleFileNameEx on your own process?

devblogs.microsoft.com/oldnewthing/20160310-00

March 10, 2016

Raymond Chen

In response to my discussion of why you can get an `ERROR_INVALID_HANDLE` from `GetModuleFileNameEx` even though the process handle is valid, Joshua asks, "Calling such methods as these on your own process has no such caveats, right?"

Well, one of the issues is that the process you are querying from hasn't yet completed its initialization. That's not an issue here, because the call is coming from within the process itself. (If it weren't initialized, then your code wouldn't be running.)

Another issue is that "the process you are inspecting may be in the middle of updating its module table, in which case the call may simply fail with a strange error like `ERROR_PARTIAL_COPY`." That issue doesn't go away just because you're making the call from within the process. Another thread in the process might be in the middle of a `LoadLibrary` call, and the module manager is adding a new entry to its table to track the new module. If you try to inspect the module table while it is being updated, you might see a partially-loaded module, you might see a linked list that is temporarily corrupt (because it is in the middle of being rewritten), you might get a corrupt string (because the file name length was copied to the entry, but the characters of the file name haven't been copied yet), or you might get an access violation (the pointer to the string hasn't been initialized yet). Any of those things can result in `GetModuleFileNameEx` failing, even when called on its own process.

Fortunately, there's a solution: Don't use `GetModuleFileNameEx` to get information about your own process. Just use the regular `GetModuleFileName` function. This function queries information for the current process, and since it always runs in-process, it can use critical sections and other synchronization objects so that it can gain access to the shared information, making sure that nobody is modifying the data structures while it is reading them.

As noted in the original article, "These APIs don't really work by the normally-accepted definitions of 'work'." They are best-effort, and sometimes the best effort fails.

Raymond Chen

Follow