The efficiency of ordinal-based imports while still being name-based

devblogs.microsoft.com/oldnewthing/20060728-00

July 28, 2006

Raymond Chen

Reader Tom brought up the interesting point that ordinal-based imports are slightly faster than name-based, though not by much. But if even that tiny fraction of a percentage bothers you, you can still get the benefits of ordinal-based imports while still being name-based. People are more familiar with the first half of the "rebase and bind" duo than the second. But it's binding that speeds up import table resolution. When a DLL is "bound" to another DLL, information about the target DLL is cached in the original DLL. Specifically, the timestamp of the target (so that the loader can detect whether the cache is valid), the ordinal corresponding to the name (the "hint"), and the address of the ultimate function. For example, if I had a DLL that linked to `kernel32!LocalAlloc` the entry in the DLL would go something like this: "Hello. I would like to link to these functions in `kernel32`. Oh, and by the way, all the hints I'm about to give you are based on the Aug 04 00:56:36 2004 version of `KERNEL32.DLL`. As for the function `LocalAlloc`, I believe that the function resides at address 0x7C8099BD, and that you'll find it in `kernel32`'s named export table in position 247."

When the loader goes to resolve the import, it checks the timestamp of the target file on the computer with the one cached in the DLL. If they match, then it doesn't need to do any look-ups at all; it justs uses the cached value (0x7C8099BD). If they don't match (for example, maybe there was a `kernel32` hot-fix), it can still use the look-up hint: Before doing the binary search for `LocalAlloc`, it looks directly at slot 247 to see if `LocalAlloc` is there. If so, then the cost of the binary search has been avoided, and the overhead of the look-up over a pure ordinal import is just one string comparison.

Raymond Chen

Follow