How can I tell that somebody used the MAKEINTRESOURCE macro to smuggle an integer inside a pointer?

devblogs.microsoft.com/oldnewthing/20130925-00

September 25, 2013

Raymond Chen

Many functions and interfaces provide the option of passing either a string or an integer. The parameter is formally declared as a string, and if you want to pass an integer, you smuggle the integer inside a pointer by using the `MAKEINTRESOURCE` macro. For example, the `FindResource` function lets you load an resource specified by integer identifier by passing the identifier in the form `MAKEINTRESOURCE(ID)`. You can tell that it was the resource-loading functions who created the macro in the first place, since the name of the macro is "make integer *resource*." But other functions use the `MAKEINTRESOURCE` convention, too. The `GetProcAddress` function lets you obtain a function exported by ordinal if you smuggle the ordinal inside a pointer: `GetProcAddress(hModule, MAKEINTRESOURCEA(ordinal))`. (You have to use `MAKEINTRESOURCEA` because `GetProcAddress` explicitly takes an ANSI string.) What if you're implementing a function whose interface requires you to accept both strings and integers-smuggled-inside strings? For example, maybe you're implementing `IContextMenu::InvokeCommand`, which needs to look at the `CMINVOKECOMMANDINFO.lpVerb` member and determine whether the invoker passed a string or a menu offset. You can use the `IS_INTRESOURCE` macro. It will return non-`FALSE` if the pointer you passed is really an integer smuggled inside a pointer.

How does `MAKEINTRESOURCE` work? It just stashes the integer in the bottom 16 bits of a pointer, leaving the upper bits zero. This relies on the convention that the first 64KB of address space is never mapped to valid memory, a convention that is enforced starting in Windows 7.

Raymond Chen

Follow