PE resources must be 4-byte aligned, but that doesn't stop people from trying other alignments

devblogs.microsoft.com/oldnewthing/20110609-00

June 9, 2011

Raymond Chen

Resources in PE-format files must be stored at offsets which are a multiple of four. This requirement is necessary for platforms which are sensitive to data alignment. That doesn't stop people from breaking the rules anyway. After all, it sort of works anyway, as long as you're careful. I mean, sure maybe if somebody running a non-x86 version of Windows tries to read your resources, they will crash, but who uses non-x86 versions of Windows, right? In Windows Vista SP1, additional hardening was added to the resource parsing code to address various security issues, but the one that's important today is that tests were made to verify that the data were properly aligned before accessing it. This prevents a file with a misaligned version resource from crashing any program that tried to read its resources. In particular, it is common for programs to read the version resources of arbitrary files—for example, Explorer does it when you view the file's properties or if you turn on the Description column in Details view—so enforcing alignment on resources closes that avenue of remote denial of service. And then the bug reports came in. "Program XYZ fails to install" because the program tries to read its own version resources and cannot, because the tool they used to build the program cheated on the alignment requirement and stored the resources at offsets that aren't multiples of 4. "I mean, come on, that wastes like three bytes per resource. Everything still worked when we removed the alignment padding, so we went ahead and shipped it that way." Another example of a program that stopped working when the alignment rules were enforced was a computer game expansion pack which could not install because the code that tried to verify that you had the base game found itself unable to read its version resources. Multiple programs refused to run, preferring to display the error message "AppName is not a valid Win32 application." Presumably, as part of initialization, they tried to read their own version resources, which failed with `ERROR_BAD_EXE_FORMAT`, which they then showed to the user. The fix was to relax the enforcement of the rules back to the previous level, and impose the stricter requirements only on architectures which raise exceptions on misaligned data. It does mean that you can have a program whose resources can be read on one machine but not on the other, but that was deemed a lesser evil than breaking all the programs which relied on being able to misalign their data without consequence.

[Raymond is currently away; this message was pre-recorded.]

Raymond Chen

Follow