Question about COFF deprecation

Posted by Will Steele - 30 Dec 2009 - 16:30

On page 11 of WinDbg_A_to_Z you mentioned that "COFF and CodeView are considered deprecated." As someone new to studying debugging and binary file structures, would there be something new to study? I have looked at some older articles (http://support.microsoft.com/kb/121460 and a few of Matt Pietrek's articles: http://msdn.microsoft.com/en-us/library/ms809762.aspx, http://msdn.microsoft.com/en-us/magazine/cc301805.aspx, http://msdn.microsoft.com/en-us/magazine/cc301808.aspx) discussing the basic structures and was unaware COFF was outdated. I know that a static analysis of a PE is not the same thing as debugging, but, thought a good understanding of PE structure would help with understanding debugging. Does studying a unloaded PE binary help much in the process? Again, my questions might be missing something obvious due to my newness to the subject.

==========

Re:Question about COFF deprecation

Posted by Robert Kuster - 01 Jan 2010 - 12:52

Hi Will,

first of all I wish you a happy and prosper 2010. May there be lot of www.windbg.info visits and forum posts. As to your questions, there are two things to distinguish:

1) Portable Executable (PE) format (used for EXE, DLL, SYS and OBJ files)

From Wikipedia: "The Portable Executable (PE) format is a file format for executables, object code, and DLLs, used in 32-bit and 64-bit versions of Windows operating systems."

The PE headers and structures are read by the Windows loader (ntdll) - it contains all the information needed to spawn a new process. Note that the PE format hasn't changed over all these years. The most obvious reason is backward compatibility. Say, if Windows 7 wants to execute an EXE written at Windows 98 times the PE format should be the same. Sure, Microsoft added a some new fields to the PE headers over the years. Nevertheless all old fields and data structures of the initial PE format are still the same. In other words you can read any old PE article and rest assured that the information provided there is valid.

Probably you won't often access PE headers during your debugging scenarios directly. But if you want my opinion it is always a good idea to understand what is going on behind the scenes. For instance, did you know that the only difference between an EXE and a DLL file is a single bit in the PE header? So yes, go on and study the available PE articles you mentioned.

One more cite from Wikipedia: "PE is a modified version of the Unix COFF file format."
Few people now days know that in its early days Microsoft had its very own version of the UNIX operation system, called Xenix. This said it is obvious why PE turns out to be a derivative of the Unix COFF file format.

2) Format of debug-symbol files (COFF, CodeView, PDB)

While data stored in the PEs is mainly intended for use by the Windows loader, a debugger actually needs more information to provide human readable information to us. This is where symbol files come in handy. Symbol files help a debugger to map raw addresses in the PE executable to source-code lines, to analyze internal layout and data of applications, and so on. Obviously symbol files are not limited with backward compatibility issues mentioned for PE files above; put it this way: with new compilers there were simply new debuggers made available.

While Microsoft used COFF and later CodeView as their official symbol files in the past, it now days uses the proprietary PDB file format to achieve the same.

It all turns out very simple, doesn't it? :)

Warm Regards, RK

==========