Don’t name your DLL “Security.dll”

devblogs.microsoft.com/oldnewthing/20040702-00

July 2, 2004

Raymond Chen

Reaching back into the history bucket... Some people have discovered that strange things happen if you name your DLL “security.dll”. The reason is that there is already a system DLL called “security.dll”; it’s the Security Support Provider Interface DLL, and it used to go by the name “security.dll”, though nowadays the name “secur32.dll” is preferred. If you look into your system32 directory, you’ll see both “security.dll” and “secur32.dll” in there. And if you’re handy with an export dumper, you’ll see that “security.dll” is just a bunch of forwarders to “secur32.dll”. If you browse through the MSDN documentation, you’ll see that everybody talks about “secur32.dll” and hardly any mention is made of its doppelgänger “security.dll”. Okay, here’s where the history comes in. Wind back to Windows 95. Back in those days, the Security Support Provider Interface was implemented in two different DLLs. The one you wanted depended on whether you are running Windows NT or Windows 95. On Windows 95, it was called “secur32.dll”, but on Windows NT, it was called “security.dll”. This was obviously a messed-up state of affairs, so the Windows NT folks decided to “go with the flow” and rename their security DLL to “secur32.dll”. This was probably for application compatibility reasons: Applications that were written to run on Windows 95 and were never tested on Windows NT just went straight for “secur32.dll” instead of loading the correct DLL based on the operating system. Okay, so now pop back to the present day. When you put a DLL called “Security.dll” in your application directory, what happens? Recall that the rules for the order in which DLLs are searched for checks the application directory before it checks the system directory. As a result, anybody in your application who wants “Security.dll” will get your version instead of the system version. Even if the system version is the one they really wanted. That’s why overriding the system’s Security.dll with your own results in a bunch of SSPI errors. Components you are using in your program are trying to talk to SPPI by loading “security.dll” and instead of getting the system one, they get yours. But yours was never meant to be a replacement for “security.dll”; it’s just some random DLL that happens to have the same name.

You would have had the same problem if you happened to name your DLL something like “DDRAW.DLL” and some component in your program tried to create a DirectDraw surface. “Security.dll” has the disadvantage that it has a simple name (which people are likely to want

to name their own DLL), and its importance to proper system functionality is not well-known. (Whereas it would be more obvious that creating a DLL called “kernel32.dll” and putting it in your application directory is going to cause nothing but trouble.)

Raymond Chen

Follow