Was there a problem with Windows 95-era programs relying on undocumented information disclosure stuff?

devblogs.microsoft.com/oldnewthing/20171107-00

November 7, 2017

Raymond Chen

Tihiy noted that back in the Windows 95 days, there was a lot of undocumented stuff lying around in places that are formally undefined. For example, the return value of `IsWindow` is formally zero or nonzero, but it turns out that on Windows 95, the nonzero value actually was a pointer to an internal data structure. I remember another case where a function returned a value in `EAX`, but it so happened that the `ECX` register contained a pointer to some interesting data structure.

These bonus undocumented values were not intentional. In the case of `IsWindow`, it was an optimization: Since the only meaningful values are zero and nonzero, and a null pointer is zero and a non-null pointer is nonzero, it was a clever trick to just return the pointer cast to an integer. In the case of the function that returned a value in `ECX`, that was completely unintentional: It was just a value that the compiler left in the `ECX` register by happenstance.

Did these undocumented but potentially useful values cause trouble?

Surprisingly not.

I'm not sure why this was not generally a problem. My guess is that software developers kept one eye on that other version of Windows, Windows NT. Relying on undocumented values wouldn't work on Windows NT, so the developers had to come up with something that would work on both.

I'm sure there were plenty of software developers who simply never tested on Windows NT or didn't consider Windows NT to be part of their customer base. But the number of those who exploited undocumented return values was small enough that I barely remember them.

Bonus chatter: I do remember one customer some time around Windows 8 who asked why the contents of the `EBX` register no longer contained a copy of the executable's instance handle when the executable entry point as called. We were kind of baffled by this question, because the contents of the `EBX` register at the executable entry point are formally undefined. Indeed, the code never explicitly sets `EBX` to anything. The value in `EBX` is

whatever the compiler happened to be using the `EBX` register for. There was no intentional effort to put a particular value into the `EBX` register. It just so happened that the instance handle was something the compiler decided to put into the `EBX` register for 19 years, and then in year 20, it decided to put something else there.

Bonus bonus chatter: You also shouldn't sniff the return address to determine how your module was loaded. That's not part of the API contract either.

Raymond Chen

Follow