# Asking questions where the answer is unreliable anyway

devblogs.microsoft.com/oldnewthing/20041115-00

November 15, 2004

Raymond Chen

Here are some questions and then explanations why you can't do anything meaningful with the answer anyway even if you could get an answer in the first place.

**"How can I find out how many outstanding references there are to a shared memory object?"**
Even if there were a way to find out, the answer you get would be instantly wrong anyway because the microsecond after you ask the question, somebody can open a new handle. This is an example of "Meaningless due to unavoidable race condition."

**"How can I find out whether a critical section is free without entering it?"**
Again, once you get an answer, the answer could instantly become wrong if another thread decides to enter the critical section immediately after you checked that it was free.

**"How can I tell whether there is a keyboard hook installed in the system?"**
This suffers from the same problem yet again: The instant you get the answer ("all clear"), somebody can install a hook.
This is actually even worse because people who ask this question are typically interested in secure keyboard access. But if somebody has a keyboard hook installed, that means that they have already injected code into your process (namely, the hook itself). At which point they could easily patch the imaginary `IsKeyboardHooked()` function to always return `FALSE`.

Now when your program asks if the keyboard is hooked, the answer is a happy "no" and you proceed, blithely confident that there are no hooks. Just because somebody said so.

**You cannot reliably reason about the security of a system from within the system itself.**

It's like trying to prove to yourself that you aren't insane.

The system may itself have already been compromised and all your reasoning therefore can be virtualized away. Besides, your program could be running inside a virtual PC environment, in which case the absence of a keyboard hook inside the virtual PC proves nothing. The keyboard logging could be happening in the virtual PC host software.

From a UI standpoint, the desktop is the security boundary. Once you let somebody run on your desktop, you implicitly trust them. Because now they can send your program random messages, inject hooks, hack at your window handles, edit your menus, and generally party all over you.

That's why it is such a horrible mistake to let a service interact with the desktop. By joining the interactive desktop, you have granted trust to a security context you should not be trusting. Sure, it lets you manipulate objects on that desktop, but it also lets the objects on that desktop manipulate you. (There's a Yakov Smirnoff joke in there somewhere, but instead I will quote Nietzsche: *Wenn du lange in einen Abgrund blickst, blickt der Abgrund auch in dich hinein*.)

If you're a service, you don't want to start letting untrusted programs manipulate you. That opens you up to a Shatter attack.

Raymond Chen

**Follow**