I brought this process into the world, and I can take it out!

devblogs.microsoft.com/oldnewthing/20120924-00

September 24, 2012

Raymond Chen

Clipboard Gadget wants to know why normal processes can kill elevated processes via `TerminateProcess`, yet they cannot do a trivial `ShowWindow(hwnd, SW_MINIMIZE)`. "Only explorer seems to be able to do so somehow." There are several questions packed into this "suggestion." (As it happens, most of the suggestions are really questions; the suggestion is "I suggest that you answer my question." That's not really what I was looking for when I invited suggestions, but I accept that that's what I basically get.) First, why can normal processes kill elevated processes? The kernel-colored glasses answer is "because the security attributes for the process grants the logon user `PROCESS_TERMINATE` access." Of course, that doesn't really answer the question; it just moves it to another question: Why are elevated processes granting termination access to the logged-on user? I checked with the security folks on this one. The intent was to give the user a way to terminate a process that they elevated without having to go through another round of elevation. If the user goes to Task Manager, right-clicks the application, and then selects "Close", and the application doesn't respond to `WM_CLOSE`, then the "Oh dear, this isn't working, do you want to try harder?" dialog box would appear, and if the user says "Go ahead and nuke it," then we should go ahead and nuke it. Note that this extra permission is granted only if the process was elevated via the normal elevation user interface (which nitpickers will point out may not actually display anything if you have enabled silent elevation). The user was already a participant in elevating the process and already provided the necessary credentials to do so. You might say that elevating a process pre-approves it for being terminated. As Bill Cosby is credited with saying, "I brought you into this world, and I can take you out!" If the process was elevated by some means other than the user interface (for example, if it was started remotely or injected by a service), then this extra permission is not granted (because it is only the elevation user interface that grants it), and the old rules apply. Phew, that's part one of the question. Now part two: Why can't you do a trivial `ShowWindow(hwnd, SW_MINIMIZE)`? Because that runs afoul of User Interface Privilege Isolation, which prevents low-integrity processes from manipulating the user interface of higher-integrity processes. My guess is that Clipboard Gadget though that terminating a process is a higher-privilege operation than being able to manipulate it. It isn't. Terminating a process prevents it from doing anything, which is different from being able to make it do anything you want. You might hire a chauffeur to drive you all over town in a

limousine, and you can fire him at any time, but that doesn't mean that you can grab the wheel and drive the limousine yourself. Finally, Clipboard Gadget wants to know how Explorer can minimize windows. Explorer does not call `ShowWindow(hwnd, SW_MINIMIZE)` to minimize windows, because Explorer is running at medium integrity and cannot manipulate the windows belonging to high-integrity processes. Instead, it posts a `WM_SYSCOMMAND` with the request `SC_MINIMIZE`. This does not minimize the window; it is merely a request to minimize the window. The application is free to ignore this request; for example, the application may have disabled its Minimize box. Most applications, however, accede to the request by minimizing the window. Just like how most chauffeurs will agree to take you to your destination along the route you specify. Unless your instructions involving going the wrong way down a one-way street or running over pedestrians.

But don't fool yourself into thinking that you're driving the limousine.

Raymond Chen

Follow