Can an application block calls to SendInput?

Asked 6 years, 10 months ago Active 4 years, 3 months ago Viewed 2k times

1

2

There's a game that I'm trying to automate some actions on.

I've used SendInput in the past very successfully. However, with this application I can't get the mouse click to work. I've tested it using other applications and it all works as expected.

Can applications block my use of SendInput? And if so, can I get around it somehow?

Side note: I'm writing code in C# and running on Windows 7 x64. The app I'm trying to interact with is x86. I don't know if this makes a difference? I've testing my code interacting with both x64 and x86 apps.

winapi automation sendinput

asked Oct 15 '13 at 13:44

Will Calderwood

3,658 2 29 52

1 Answer

Active | Oldest | Votes

3

Short answer: No. (Not the call to `SendInput`, but the input can be filtered. See update below.)

If you look at the parameters for `SendInput` there is nothing that identifies a process. The input is sent to the system, not an application. An application has no way of telling the difference between real and synthesized input.

There are a number of reasons why an application will not respond to synthesized input. As explained in the documentation for `SendInput` this API is subject to UIPI. An application running at a higher integrity level than an application calling `SendInput` will not receive this input.

Although `SendInput` injects input at a lower level than DirectInput runs, DirectInput is apparently more susceptible to buggy code. See Simulating Keyboard with SendInput API in DirectInput applications for reference.

Update (2016-05-01):
Besides issues with UIPI, preventing input from reaching an application, it is also possible for a low-level keyboard/mouse hook to identify injected input. Both the KBDLLHOOKSTRUCT (passed to the LowLevelKeyboardProc callback) as well as the MSLLHOOKSTRUCT (passed to the LowLevelMouseProc callback) contain a *flags* member, that have the `LLKHF_INJECTED` or `LLMHF_INJECTED` flag set, in case the input is injected.

An application can thus install a low-level keyboard/mouse hook to filter out messages that are injected. If this is the case, a potential workaround (without writing a keyboard driver) is, to

installing itself to the top of the hook chain, this rendering the workaround useless.

edited May 23 '17 at 12:07
Community ♦
1 1

answered Oct 15 '13 at 14:10
IInspectable
32.1k 6 59 132

Thanks for the response. I assume if I run as administrator, that would get round any UIPI problem? If that's the case, then I can discount that. – Will Calderwood Oct 15 '13 at 14:35