

Access OSK(on screen keyboard) handle from desktop app without administrator rights

Asked 9 years, 10 months ago Modified 7 years, 11 months ago

Viewed 3k times



0



I have a desktop delphi application that runs without administrator rights on windows 7 and 8. This application, needs to send (SendInput) mouse events(click and move) to another running apps. This app works like a driver for a remote wifi pen, that controls mouse over desktop. When the focus is over OSK(on screen keyboard), the mouse move with left key pressed dont work, the osk windows dont move, all others applications move when receive these mouve events. I cand get handle of OSK. When I run my app with administrator privileges(UAC) all works fine, OSK move when app send mouve envets. I think that problem is related to UAC. I found a way to bypass the UAC like this <http://www.thewindowsclub.com/create-elevated-shortcut-run-programs-bypass-uac>, but is not a good ideia in some enviroments. There is a way to bypass the UAC without underground ways ? Or how can I force the OSK to respond on all mouse events that I send to him.

[windows](#)[winapi](#)[privileges](#)[kiosk-mode](#)[Share](#)[Improve this question](#)[Follow](#)

edited Feb 11, 2015 at 12:44



David Heffernan

612k ● 43 ● 1.1k ● 1.5k

asked Feb 11, 2015 at 12:36



Leonardo

159 ● 2 ● 17

1 Answer

Sorted by:

Highest score (default)



4



Here is a snip of the manifest that is embedded in Osk.exe:

```
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel level="asInvoker" uiAcc
    </requestedPrivileges>
    </security>
  </trustInfo>
```

Note the `level` it asks for, *asInvoker* does **not** ask for UAC elevation, only *requireAdministrator* does. In other words, it runs with whatever privileges the starting program has. You can tell, you don't get the consent prompt when you start Osk.exe

What matters here is `uiAccess`. With it set to *true*, the program bypasses UIPI. The lesser-known twin of UAC,

User Interface Privilege Isolation protects against shatter attacks by disallowing another process to poke keystrokes and mouse clicks into the window owned by an elevated app. Such a process still runs in high integrity mode, that's why you cannot poke into Osk yourself, but doesn't have the privileges enabled that make an UAC elevated app dangerous.

This is not unusual, most any program that uses UI Automation or provides an accessibility feature needs to be able to do this. Like Osk.exe, it needs to be able to poke keystrokes into any app. Clearly what you want to do as well.

Getting uiAccess does not require the user to consent to a prompt like UAC elevation does. The operating system has to "trust" you. Covered well in [this MSDN article](#), "UIAccess for UI automation applications" section. I'll just summarize it here:

- Set uiAccess="true" in the application manifest
- Your executable must have valid digital certificate, the kind you buy from a vendor like Verisign.
- Your executable must be stored in a directory that has write access denied, in a subdirectory of c:\program files or c:\windows.

Share Improve this answer

answered Feb 11, 2015 at 14:01


Follow




Hans Passant

940k ● 148 ● 1.7k ● 2.6k

-
- 1 Do note, however, that `SendInput()` is not poking anything into OSK and vice versa. `SendInput()` posts input messages into the same queue that the hardware keyboard/mice do. By the time the input messages are processed by the OS, whatever app is currently in focus receives them. So OSK (or any other app) does not know or care whether mouse clicks are fake or real. Raymond Chen has a [good description about that on his blog](#)...
- Remy Lebeau Feb 12, 2015 at 0:34
-

- 1 ..., so your explanation, while useful and informative, does not really explain why OSK is reacting to mouse input from `SendInput()` only when Leonardo's app is running elevated. It should always be reacting regardless of whether the app is running elevated or not. UIPI avoids attacks from messages being sent directly to a window across different integrity levels. That is not the case with `SendInput()`. If anything, when Leo's app is not running elevated, it is likely running at an integrity level equal to OSK and would not be blocked by UIPI anyway, unless OSK is running elevated.
- Remy Lebeau Feb 12, 2015 at 0:35 
-

Which now makes me wonder if OSK is actually being run elevated in Leo's case. If it were, that MIGHT explain why he can't control it from a non-elevated process, if this really did turn out to be a UAC/UIPI issue. – Remy Lebeau Feb 12, 2015 at 0:41 

Well, the solution with `uiAccess=true` of Hans Passant works without prompt of elevation like suggested. In a simple app named `uac.exe`, the procedure of hans works, I can move the `osk` window. But on final app, I do the same process and pops up an alert with "Invalid file name", all is the same, the only difference is the name of fail app that has 9 chars of length, `PenDriver.exe`. Have limitation of 8 in manifest ?

– Leonardo Feb 12, 2015 at 14:33
