AttachThreadInput is like taking two threads and pooling their money into a joint bank account, where both parties need to be present in order to withdraw any money

devblogs.microsoft.com/oldnewthing/20130619-00

June 19, 2013

Raymond Chen

Consider this code:

```
// Code in italics is wrong
   foregroundThreadId = ::GetWindowThreadProcessId(::GetForegroundWindow(), 0);
   myThreadId = GetCurrentThreadId();


   if (foregroundThreadId != myThreadId)
   {
       AttachThreadInput(foregroundThreadId, myThreadId, TRUE);
       BringWindowToTop(myWindowHandle);
```

If you try to step over the `AttachThreadInput` call in the debugger, both the debugger and the application being debugged will freeze. Why is that?

This should look familiar because it's basically the same code that I warned you about several years ago. The code grabs the current foreground window and attaches its input state to the current thread. Now you're in trouble.

Remember dual-signature bank accounts? These were bank accounts that required the signatures of both account holders in order to make a withdrawal. It can work out fine if the two parties trust each other with a shared bank account and can coordinate their actions so that when one of them needs money, it can go to the other and say, "Hey, can you sign this withdrawal slip? I need some money." (Another use case for dual-signature bank accounts was a parent wanting to monitor their child's spending.)

`AttachThreadInput` tells the window manager, "Please take these two threads and put all their money in a dual-signature bank account."

In the case above, the code said, "See that *random person being served by the bank teller*? Please take all my money and all his money and put them into a dual-signature bank account."

As you can imagine, this is a bad idea, both for you and for the other person. You cannot withdraw any money until you can somehow track down that random person and get him to sign the withdrawal form. And it's not like you have any relationship with that person—you don't even know his name!—so the only chance you have is to go down to the bank and hang out there hoping that the other guy will show up to make a withdrawal as part of his normal course of business, and then you can say, "Hey, you there! Sign this for me, will ya?"

The other person is in a similar predicament. When he goes to the bank to make a withdrawal, the teller will say, "I'm sorry, sir, but your money is in a dual-signature account, and your withdrawal slip has only one signature on it." He's stuck doing the same thing that you do: Whenever he wants to withdraw money, he has to go to the bank and hang around hoping that you will show up eventually.

bank account	↔	input queue
money	↔	input
go to the bank	↔	check the message queue

In this case, what happened was that the code grabbed the debugger and said, "Okay, we now have a dual-signature bank account!" And now you're stuck. The debugger cannot withdraw any money because it is waiting for you to go to the bank. But you can't go to the bank because you're broken into the debugger. Result: Nobody gets any money.

This is why you shouldn't grab random people in the bank and unilaterally create dual-signature bank accounts with them.

Reminder: Attaching input queues is not a Get Out of Jail Free card. It's a *Get Into the Same Jail* card.

Raymond Chen

Follow