How can I launch an unelevated process from my elevated process and vice versa?

devblogs.microsoft.com/oldnewthing/20131118-00

November 18, 2013

Raymond Chen

Going from an unelevated process to an elevated process is easy. You can run a process with elevation by passing the `runas` verb to `ShellExecute` or `ShellExecuteEx`.

Going the other way is trickier. For one thing, it's really hard to munge your token to remove the elevation nature properly. And for another thing, even if you could do it, it's not the right thing to do, because the unelevated user may be different from the elevated user.

Let me expand on that last bit.

Take a user who is not an administrator. When that user tries to run a program with elevation, the system will display a prompt that says, "Hey, like, since you're not an administrator, I need you to type the userid and password of somebody who *is* an administrator." When that happens, the elevated program is running not as the original user but as the administrative user. Even if the elevated program tried to remove elevation from its token, all it managed to do is create an unelevated token *for the administrative user*, not the original user.

Suppose we have Alice Administrator and Bob Banal. Bob logs on, and then tries to run LitWare Dashboard, which requires elevation. The prompt comes up, and Bob calls over Alice to grant administrative privileges. Alice types her password, and boom, now LitWare Dashboard is running elevated *as Alice*.

Now suppose LitWare Dashboard wants to launch the user's Web browser to show some online content. Since there is no reason for the Web browser to run elevated, it tries to unelevate the browser in order to reduce the security attack surface. If it simply neutered its token and used that to launch the browser, it would be running a copy of the browser unelevated *as Alice*. But LitWare Dashboard presumably really wanted to run the browser as Bob, since it is Bob who is the unelevated user in this session.

The solution here is to go back to Explorer and ask Explorer to launch the program for you. Since Explorer is running as the original unelevated user, the program (in this case, the Web browser) will run as Bob. This is also important in the case that the handler for the file you want to open runs as an in-process extension rather than as a separate process, for in that case, the attempt to unelevate would be pointless since no new process was created in the first place. (And if the handler for the file tries to communicate with an existing unelevated copy of itself, things may fail because of UIPI.)

Okay, I know that Little Programs are not supposed to have motivation, but I couldn't help myself. Enough jabber. Let's write code. (Remember that Little Programs do little or no error checking, because that's the way they roll.)

```
#define STRICT
#include <windows.h>
#include <shldisp.h>
#include <shlobj.h>
#include <exdisp.h>
#include <atlbase.h>
#include <stdlib.h>
// FindDesktopFolderView incorporated by reference
void GetDesktopAutomationObject(REFIID riid, void **ppv)
{
 CComPtr<IShellView> spsv;
 FindDesktopFolderView(IID_PPV_ARGS(&spsv));
 CComPtr<IDispatch> spdispView;
 spsv->GetItemObject(SVGIO_BACKGROUND, IID_PPV_ARGS(&spdispView));
 spdispView->QueryInterface(riid, ppv);
}
```

The `GetDesktopAutomationObject` function locates the desktop folder view then asks for the dispatch object for the view. We then return that dispatch object in the form requested by the caller. This dispatch object is a `ShellFolderView`, and the C++ interface for that is `IShellFolderViewDual`, so most callers are going to ask for that interface, but if you are a masochist, you can skip the dual interface and talk directly to `IDispatch`.

```
void ShellExecuteFromExplorer(
    PCWSTR pszFile,
    PCWSTR pszParameters = nullptr,
    PCWSTR pszDirectory  = nullptr,
    PCWSTR pszOperation  = nullptr,
    int nShowCmd         = SW_SHOWNORMAL)
{
 CComPtr<IShellFolderViewDual> spFolderView;
 GetDesktopAutomationObject(IID_PPV_ARGS(&spFolderView));
 CComPtr<IDispatch> spdispShell;
 spFolderView->get_Application(&spdispShell);
 CComQIPtr<IShellDispatch2>(spdispShell)
    ->ShellExecute(CComBSTR(pszFile),
                   CComVariant(pszParameters ? pszParameters : L""),
                   CComVariant(pszDirectory ? pszDirectory : L""),
                   CComVariant(pszOperation ? pszOperation : L""),
                   CComVariant(nShowCmd));
}
```

The `ShellExecuteFromExplorer` function starts by getting the desktop folder automation object. We use the desktop not because it's particularly meaningful but because we know that it's always going to be there.

As with the desktop folder view, the `ShellFolderView` object is not interesting to us for itself. It's interesting to us because the object resides in the process that is hosting the desktop view (which is the main Explorer process). From the `ShellFolderView`, we ask for the `Application` property so that we can get to the main `Shell.Application` object, which has the `IShellDispatch` interface (and its extensions `IShellDispatch2` through `IShellDispatch6`) as its C++ interfaces. And it is the `IShellDispatch2::ShellExecute` method that is what we really want.

"You never loved me. You only wanted me in order to get access to my family," sobbed the shell folder view.

And we call `IShellDispatch2::ShellExecute` with the appropriate parameters. Note that the parameters to `IShellDispatch2::ShellExecute` are *in a different order* from the parameters to `ShellExecute`!

Okay, let's put this inside a little program.

```
int __cdecl wmain(int argc, wchar_t **argv)
{
 if (argc < 2) return 0;
 CCoInitialize init;
 ShellExecuteFromExplorer(
    argv[1],
    argc >= 3 ? argv[2] : L"",
    argc >= 4 ? argv[3] : L"",
    argc >= 5 ? argv[4] : L"",
    argc >= 6 ? _wtoi(argv[5]) : SW_SHOWNORMAL);
 return 0;
}
```

The program takes a mandatory command line argument which is the thing to execute, be it a program or a document or a URL. Optional parameters are the parameters to the thing being executed, the current directory to use, the operation to perform, and how the window should be opened.

Open an elevated command prompt, and then run this program in various ways.

`scratch http://www.msn.com/`	Open an unelevated Web page in the user's default Web browser.
`scratch cmd.exe "" C:\Users "" 3`	Open an unelevated command prompt at `C:\Users`, maximized.
`scratch C:\Path\To\Image.bmp "" "" edit`	Edit a bitmap in an unelevated image editor.

This program is basically the same as the Execute in Explorer sample.

Raymond Chen

Follow