Local elevation points in Windows and Delphi (http://www.sandon.it/node/90)

Posted on: Fri, 05/16/2014 - 16:42 **By:** ldsandon

Since the introduction of Windows Vista and the new security model for applications, application running under User Account Control (UAC) should adopt a "least privilege" model, running as an "unprivileged" user almost all the time, and requesting higher privileges only when needed, even if the user has those privileges.

Requesting higher privileges is called "elevation". A good application uses "local elevation points", meaning it elevates only when it really needs it, and then reverts to a non elevated stated afterwards. These operations are those identified by a little shield on the control (button, menu item, etc.) that activates them.

But how to perform this kind of elevation? There is not a simple way, say an ElevateProcess() or ElevatedThread() API. First, elevation can't be performed for a single thread. It needs to be performed at the process level, and there are good security reason behind this choice. Second, elevating a whole process would also elevate all threads within. Thereby, elevation require to "spawn" a new process. There are at least three different ways to perform this, in this post I'll explain what I believe is the most elegant and flexible one, albeit complex - the COM Elevation Moniker.

Through a COM Elevation Moniker we can "spawn" a COM local server running with elevated privileges, and obtain an interface to perform the operations we need. They are actually carried out by the interface implementation in the COM process, and thereby isolated by the calling process. COM does all the checks and UI management we need, displaying the proper UI dialogs to confirm (or cancel) elevation, and asking for credential if needed. What we have to implement is the COM server, and register it so it can be called elevated.

Once the server is ready, the key call is a function alike this (this is my Delphi translation of the example you can find on MSDN (http://msdn.microsoft.com/en-us/library/ms679687.aspx), this implementation was written in XE2 and thereby assumes Unicode strings):

```
<br />
function CoCreateInstanceAsAdmin(hwnd: HWND; rclsid: TCLSID; riid:
 TIID; out ppv: Pointer): HRESULT;<br />
  var bo: BIND_OPTS3;<br />
  wszCLSID: POLESTR;<br />
  wszMonikerName: LPCWSTR;<br />
begin<br />
  Result := S_FALSE;<br />
  wszCLSID := nil;<br />
  wszMonikerName := nil;<br />
  try<br />
    wszCLSID := StrAlloc(50);<br />
    wszMonikerName := StrAlloc(300);<br />
    if StringFromGUID2(rclsid, wszCLSID, 50) = 0 then<br />
      RaiseLastOSError;<br />
    StrCopy(wszMonikerName, PWideChar(Format("Elevation:
Administrator!new:%s", [wszCLSID])));<br />
    FillChar(bo, SizeOf(bo), 0);<br />
    bo.cbStruct := SizeOf(bo);<br />
    bo.hwnd := hwnd;<br />
    bo.dwClassContext := CLSCTX_LOCAL_SERVER;<br />
   Result := CoGetObject(wszMonikerName, @bo, riid, ppv);<br />
  finally<br />
    StrDispose(wszCLSID);<br />
    StrDispose(wszMonikerName);<br />
  end;<br />
end;<br />
```

This function takes a class ID (rclsid), an interface ID (riid) and returns a pointer to the interface if it can instantiate the required COM class implementing that interface - you can of course cast it to the interface you requested. What allows elevation is the calling through the "'Elevation:Administrator!new:" moniker (you can also request the "Highest" run level instead of "Administrator".

You can implement a COM server in Delphi the usual way. Just remember it needs to be called as a "local server", not an "inproc" one.

Not all COM server can be elevated. They must be registered to allow it. Registration needs some additional steps (aka registry settings) in the registration procedure.

First of all, the class needs a "friendly name" that is displayed in the elevation dialog:

HKEY_LOCAL_MACHINE\Software\Classes\CLSID {CLSID} LocalizedString = displayName Where CLSID is the Class ID of the server, and displayName needs to be MUI (Multilingual User Interface) compliant. It means COM won't use the string you write here, but uses it to lookup the display name into a MUI resource. That string is usually in the form:

@<path>,-resID

and thereby you need to implement this as well. Without being able to retrive this string, Windows will return an error, either CO_E_MISSING_DISPLAYNAME, or a MUI api call error.

To enable elevation, this entry is needed:

HKEY_LOCAL_MACHINE\Software\Classes\CLSID {CLSID} Elevation Enabled = 1

Otherwise a CO_E_ELEVATION_DISABLED error is returned. MSDN says these keys must be in HKLM, to hinder a non privileged user to elevate classes they have not the privilege to register.

You can also set an icon to be dispalyed in the UAC UI, using:

HKEY_LOCAL_MACHINE\Software\Classes\CLSID {CLSID} Elevation IconReference = applicationIcon

The applicationIcon syntax is the same for LocalizedString, of course it needs to point to an icon resource :)

Because the interface implementation is running elevated, I suggest to implement it to perfrom "as little as needed" - too broad implementation may open security risks - just perform what you need and only that, don't create general calls that can do almost anything.

This kind of elevation implementation is in my opinion very clean and safe, the use of COM interface allows for compile-time checks, exception handing, etc. etc. unlike other solutions. It requires some work to implement the COM server and register it, but it's time well spent to deliver a well behaved and professionally looking application.

Just, don't forget to put the little shields where needed...

Be aware that the kind of UAC UI displayed depends on the COM server executable signature. If the executable is signed with a valid certificate, the UI is the "calm, confirmation only one". If the executable is not signed, or signed with an invalid certificate, the UI is the "warning, are you really sure?" one. For development you can generate your own certificates (I suggest XCA (http://sourceforge.net/projects/xca/) to create and manage your own PKI), but for deployed applications I recommend to obtain a real signing certificate for your applications.

- Log in (http://www.sandon.it/user/login?destination=/comment/reply/node/90/comment_node_blog%23comment-form) to post comments