Just because you're using a smart pointer class doesn't mean you can abdicate understanding what it does

devblogs.microsoft.com/oldnewthing/20080623-00

June 23, 2008

Raymond Chen

It's great when you have a tool to make programming easier, but you still must understand what it does or you're just replacing one set of problems with another set of more subtle problems. For example, we discussed earlier the importance of knowing when your destructor runs. Here's another example, courtesy of my colleague Chris Ashton. This was posted as a Suggestion Box entry, but it's pretty much a complete article on its own.

> I came across an interesting bug this weekend that I've never seen described anywhere else, I thought it might be good fodder for your blog.
>
> What do you suppose the following code does?
>
> ```
> CComBSTR bstr;
> bstr = ::SysAllocStringLen(NULL, 100);
> ```
>
> a. Allocates a `BSTR` 100 characters long.
> b. Leaks memory and, if you're really lucky, opens the door for an insidious memory corruption.
>
> Obviously I'm writing here, so the answer cannot be A. It is, in fact, B.
>
> The key is that `CComBSTR` is involved here, so `operator=` is being invoked. And `operator=`, as you might recall, does a deep copy of the entire string, not just a shallow copy of the `BSTR` pointer. But how long does `operator=` think the string is? Well, since `BSTR` and `LPCOLESTR` are equivalent (at least as far as the C++ compiler is concerned), the argument to `operator=` is an `LPCOLESTR` – so `operator=` naturally tries to use the `wcslen` length of the string, not the `SysStringLen` length. And in this case, since the string is uninitialized, `wcslen` often returns a much smaller value than `SysStringLen` would. As a result, the original 100-character string is leaked, and you get back a buffer that can only hold, say, 25 characters.
>
> The code you *really* want here is:
>
> ```
> CComBSTR bstr;
> bstr.Attach(::SysAllocStringLen(NULL, 100));
> ```
>
> Or:
>
> ```
> CComBSTR bstr(100);
> ```
>
> I'm still a big fan of smart pointers (surely the hours spent finding this bug would have been spent finding memory leaks caused by other incautious programmers), but this example gives pause – `CComBSTR` and some OLE calls just don't mix.

All I can add to this story is an exercise: Chris writes, "Since the string is uninitialized, `wcslen` often returns a much smaller value than `SysStringLen` would." Can it possibly return a *larger* value? Is there a potential read overflow here?

Raymond Chen

Follow