How am I supposed to free the information returned by the GetSecurityInfo function?

devblogs.microsoft.com/oldnewthing/20121212-00

December 12, 2012

Raymond Chen

The `GetSecurityInfo` function returns a copy of the security descriptor for a kernel object, along with pointers to specific portions you request. More than once, a customer has been confused by the guidelines for how to manage the memory returned by the function.

Let's look at what the function says:

> ***ppsidOwner*** **[out, optional]**
> A pointer to a variable that receives a pointer to the owner SID in the security descriptor returned in *ppSecurityDescriptor*. The returned pointer is valid only if you set the `OWNER_SECURITY_INFORMATION` flag. This parameter can be NULL if you do not need the owner SID.

Similar verbiage can be found for the other subcomponent parameters. The final parameter is described as

> ***ppSecurityDescriptor*** **[out, optional]**
> A pointer to a variable that receives a pointer to the security descriptor of the object. When you have finished using the pointer, free the returned buffer by calling the **LocalFree** function.

Okay, so it's clear that you need to free the security descriptor with `LocalFree`. But how do you free the owner, group, DACL, and SACL?

Read the documentation again. I've underlined the important part.

> ***ppsidOwner*** **[out, optional]**
> A pointer to a variable that receives a pointer to the owner SID in the security descriptor returned in *ppSecurityDescriptor*. The returned pointer is valid only if you set the `OWNER_SECURITY_INFORMATION` flag. This parameter can be NULL if you do not need the owner SID.

In case that wasn't clear, the point is reiterated in the remarks.

> If the *ppsidOwner*, *ppsidGroup*, *ppDacl*, and *ppSacl* parameters are non-NULL, and the *SecurityInfo* parameter specifies that they be retrieved from the object, those parameters will point to the corresponding parameters in the security descriptor returned in *ppSecurity-Descriptor*.

In other words, you are getting a pointer *into the security descriptor*. No separate memory allocation is made. The memory for the owner SID is freed when you free the security descriptor. It's like the last parameter to `GetFullPathName`, which receives a pointer to the file part of the full path. There is no separate memory allocation for that pointer; it's just a pointer back into the main buffer.

You can think of the `ppsidOwner` parameter as a convenience parameter. The `GetSecurityInfo` function offers to do the work of calling `GetSecurityDescriptorOwner` for you. You can think of the function as operating like this:

```
DWORD WINAPI GetSecurityInfo(...)
{
    ... blah blah get the security info ...
    // Just out of courtesy:
    // Fetch the owner if the caller requested it
    if (ppsidOwner != NULL &&
        (SecurityInfo & OWNER_SECURITY_INFO)) {
        BOOL fDefaulted;
        GetSecurityDescriptorOwner(pSecurityDescriptor,
                                   ppsidOwner,
                                   &fDefaulted);
    }
    ...
}
```

That's why the documentation says that you need to pass a non-null `ppSecurity-Descriptor` if you request any of the pieces of the security descriptor: If you don't, then you won't be able to free the memory for it.

Bonus chatter: If the `ppSecurityDescriptor` is so important, why is it marked "optional"?

It really should be a mandatory parameter, but older versions of Windows didn't enforce the rule, so the parameter is grandfathered in as optional, even though no self-respecting program should ever pass in `NULL`. If you pass `NULL` for the `ppSecurityDescriptor`, the function happily allocates the security descriptor and then, "Oh wait, the caller didn't give me a way to receive the pointer to the security descriptor, so I guess I won't give it to him."

```
DWORD WINAPI GetSecurityInfo(...)
{
    ... blah blah get the security info ...
    if (ppSecurityDescriptor != NULL) {
        *ppSecurityDescriptor = pSecurityDescriptor;
    }
    ...
}
```

Result: Memory leak.

You might say that the last parameter was designed by somebody wearing kernel-colored glasses.

Raymond Chen

Follow