The security check happens at the acquisition of the handle

devblogs.microsoft.com/oldnewthing/20200320-00

March 20, 2020

Raymond Chen

When you call a function like `CreateFile` or `OpenMutex`, a security check is made to ensure that you have access to the underlying object before returning a handle.

But once you get the handle, it’s yours to keep.

Changes to the security of the underlying object affect the security checks for future attempts to obtain handles to the object, but existing handles continue to have the access they have.

This behavior is a feature, not a bug.

Impersonation relies on this feature: After you impersonate a user, you can open a file that the user has access to, but any handles you opened prior to impersonation can still be used. And then after you revert impersonation, you can continue to use that handle obtained while impersonating, even though you are not impersonating any longer.

Allowing already-obtained handles to continue to be used means that programs that do impersonation can operate on objects on “both sides of the fence”: It can simultaneously access resources that are accessible only to to the original user, as well as resources that are accessible only to the impersonated user.

This technique is also used by server processes. The client requests a resource, and the server uses its server powers to open a handle to that resource, then returns the handle to the client via duplication. Thus, the client obtains access to a resource that the client itself may not have had direct access to. Since a handle is returned, the client can operate directly with the resource via that handle, which is more efficient (and less work) than having to broker every operation on the resource through the server.

Performing the security check at handle creation allows expensive work to be front-loaded. Enumerating the ACEs, matching them up against the identities in the SID, generating any requested audit entries into the security event log: That can happen just once, at the handle creation. Subsequent actions need only check the granted access mask that is recorded in the handle.

However, this behavior also means that if you change the security on a resource, the change affects only future attempts to obtain a handle to that resource. Pre-existing handles continue to have whatever level of access they already had. This means that if you, say, make a file read-only, anybody who had already obtained a write handle to the file can still write to it via that handle. They won't be able to make any new write handles, but they can use the one they already have.

If you really want to make sure nobody can write to the file any more, you need to make sure nobody got "grandfathered in". If the file is on a server and you are concerned about a client with a lingering write handle, you can use the Computer Management snap-in, go to Shared Folders, Open Files, and then close any existing handles to the file you are worried about.[1] If you are worried about local access, you can restart the computer that has the file.

[1] Note that closing handles via the snap-in is not the same as forcing handles closed by injecting a `CloseHandle` into the client process. The snap-in invalidates the handle on the server, but the handle is still valid on the client. When the client tries to use the handle, it will be told, "Sorry, that handle can no longer be used to access the resource. You'll just have to close it and get a new handle."

Raymond Chen

Follow