Why doesn't RegSetKeySecurity propagate inheritable ACEs, but SetSecurityInfo does?

devblogs.microsoft.com/oldnewthing/20200102-00

January 2, 2020

Raymond Chen

A customer was updating some code that manipulated registry key security. The old code used the `SetSecurityInfo` function to update the security, and they observed that the function propagates inheritable ACEs to child keys. Their revised version used `RegSetKey-Security`, and they observed that that function does *not* propagate inheritable ACEs to child keys.

What's going on?

Inheritable ACEs are a somewhat artificial concept. The access check occurs on the object, so all that matters is the security descriptor on the object, not on any of its parents. (For objects like files, there may not even be a unique parent, thanks to hard links.)

An inheritable ACE is an ACE in a parent object which says, "If a child object is created, copy this ACE to the child object's security descriptor." Setting an inheritable ACE lets you influence the algorithm the system uses to generate the default ACL for the new object. The inherited ACEs in the new object are marked with a flag that says "Oh, and in case anybody wants to know, this ACEs was created due to inheritance."[1]

Of course, the creator of the object may choose to override the default, but at least you got to make your wishes known, and if they chose to ignore those wishes, that's their decision.

In Windows 2000, the inheritance model for security attributes changed. This article has a comparison table.

Under the old model, the ACL for an object was calculated from the parent at the time the object was created, and that was it. Changes to the parent's inheritable attributes had no effect on the children. Programs which updated the parent's ACL typically offered a checkbox called something like "Replace permissions on child objects." If you did this, then the program recalculates the ACLs for child objects as if they had been newly-created. This recalculated ACL would pick up the new inheritable ACEs. Then the program goes through and sets that replacement ACL on all child objects.

The “Replace permissions on child objects” option was a rather crude solution which obliterated any customizations that had been applied to the children.

Under the new model, when you change the parent ACL, the security attributes of all child objects are automatically recalculated, and it is done in a more precise manner: Only the inherited ACEs are replaced. The custom attributes on child objects are preserved. This takes advantage of the “In case anybody wants to know” flag that is set when an ACE is created due to inheritance.

The `RegSetKeySecurity` function is an old function that follows the old model. It replaces the ACLs on the object and does nothing about inheritance.

The `SetSecurityInfo` function is a new function that follows the new model. It replaces the ACLs on the object and updates the inheritable ACEs for all child objects.

So the reason the two functions behave differently is that they were written at different times, when the rules for inheritable ACEs were different, and each one follows the rules in effect at the time they were written.

Bonus chatter: The same explanation applies to pairs of functions like `SetFileSecurity` (old and busted) and `SetNamedSecurityInfo` (new hotness).

[1] It is this flag that triggers the ACL editor to go look for the thing that the ACE was inherited from.

Raymond Chen

Follow