

Strange behavior when setting ACL on NTFS Folder

Source:

<http://www.tech-archive.net/Archive/Scripting/microsoft.public.scripting.vbscript/2004-03/1636.html>

From: Yuri Niyazov (*yuricakespam-google_at_yahoo.com*)

Date: 03/25/04

Date: 25 Mar 2004 13:28:11 -0800

I am experiencing the following strange behavior on NTFS, Windows 2000 Server SP3.

First, I, logged in as Administrator, manually create a folder called C:\acltest, on it I remove all inherited ACE entries and create a "Everyone" "Full Control" "This Folder Only" ACE entry on it, using the right-click properties Explorer UI.

Then, I create a "patient" folder under acltest using the first pasted below vbs script (acltest-createfolder.vbs)

Then, I have another VBS WMI script set up certain read/write permissions for on the patient folder for users. That script is acl-script.vbs, pasted below.

I execute both of these scripts at the console. I then right click on C:\acltest\patient and check the list of permissions in Properties dialog box. Everything comes out as I want it, including the "modify" entry for the user.

Now, without doing anything else, I, as the user added (in this case "yuri"), use terminal services and connect to this machine from another workstation. I try to open "C:\acltest\patient" and I receive an "Access Denied" entry – something totally unexpected, since the permissions tell me that yuri is allowed to look at the contents of the folder. Now, I go back to the console, and using the Properties dialog box add some completely different ACE entry to C:\acltest\patient – something that should have no effect on the current situation, for example, a "Guest" "Deny Delete" "Subfolders only" entry. Click OK, and now, go back to the terminal services as "yuri" and try to open C:\acltest\patient folder – and all of a sudden, it's opened.

The behavior seems to suggest that I am doing something that, even though it adds ACE entries into the Properties dialog box, it doesn't

microsoft.public.scripting.vbscript: Strange behavior when setting ACL on NTFS Folder

actually write (flush?) them out into whatever Security mechanism NTFS uses. After I click "OK" in properties dialog box, everything is probably re-written out properly.

Further information about my setup: the machine is a Active Directory Domain Controller, yuri is a member of Domain Users, and is not a member of Power Users or Administrators. I did modify "Domain Controller Security Policy" to allow common users to use terminal services to login to the Domain Controller.

Files (I am posting from google, I apologize about the wrapping. if it is a problem, email me and I will send them as files):

acltest-createfolder.vbs:

```
'START
MsgBox "Requirements: create a C:\acltest folder." & vbCrLF & "Make
sure that it is not inheriting any parent ACEs." _
  & vbCrLf & "Add on it the following entry: 'Everyone' 'Full Control'
'This Folder Only'"

set fso = CreateObject("Scripting.FileSystemObject")
fso.CreateFolder "C:\acltest\patient"
'END
```

acl-script.vbs:

```
'START
Msgbox "start"
r = SetPermissions("C:\acltest\patient", "Everyone", "DELETE", 1, 0)
a = SetPermissions("C:\acltest\patient", "yuri", "ACCESS", 0, 3)
if r = 1 or a = 1 then
  msgbox "error"
end if
Msgbox "stop"
```

#####

Function SetPermissions

'Variables

'ExplicitPath (i.e. e:\data\folder – use GetExplicitPathFromUNC
function if necessary)

'UserOrGroupName

'Permission (Read,Modify,Full)

'Returns

'0 if successful

'1 if failed

microsoft.public.scripting.vbscript: Strange behavior when setting ACL on NTFS Folder

Function SetPermissions(ExplicitPath,UserOrGroupName,Permission,AceType,AceFlags)

Found = 0

'Create the security object. You have to invoke the Security and
Restore Priviledges in the

'script to make changes to ACEs

Set objSecurity =

GetObject("winmgmts:\\.\\root\cimv2:Win32_LogicalFileSecuritySetting=""
& ExplicitPath & """)

If objSecurity.GetSecurityDescriptor(objSD) = 0 Then

Else

MsgBox "Problem"

SetPermissions = 1

Exit Function

End If

'Determine which Permission to be applied

' If ucase(Permission) = "READ" Then PermissionBitMask = 1179817

' If ucase(Permission) = "MODIFY" Then PermissionBitMask = 1245631

' If ucase(Permission) = "FULL" Then PermissionBitMask = 2032127

 If ucase(Permission) = "ACCESS" Then PermissionBitMask = 197055

 If ucase(Permission) = "DELETE" Then PermissionBitMask = 65536

'Create the Security Descriptor object (objSD) and then add, modify,
or delete ACEs inside

'the DACL.

'If Found = 1, then the specified User or Group was modified. Use

SetSecurityDescriptor

'to save the changes.

If Found = 1 Then

 If objSecurity.SetSecurityDescriptor(objSD) = 0 Then

 SetPermissions = 0

 Else

 SetPermissions = 1

 Err.Clear

 End If

Else

'Specified User or Group was NOT found in the existing DACL. Add a new
ACE to the

'DACL

'(1) Get the SID of the user or group account.

'Create a WMI object on the local computer (Assuming the local
computer is in the

'same domain (or in a trusted domain) as the one specified

 Set objWMI = GetObject("winmgmts:\\.\\root\cimv2")

Strange behavior when setting ACL on NTFS Folder

microsoft.public.scripting.vbscript: Strange behavior when setting ACL on NTFS Folder

'Get a collection of Users objects, using ExecQuery

```
Set colUsers = objWMI.ExecQuery("SELECT * FROM WIN32_ACCOUNT  
WHERE Name = '' & UserOrGroupName & ''")
```

'The collection should only have one user in it. Loop through the one user collection

'and create a user object

```
For Each UserName in colUsers  
Set objUserName = UserName  
Next
```

'Next, we need both the binary and string representation of the user's SID to create a new

Win32_Trustee. objUser.SID only returns the string version. Create a Win32_SID object by

'referencing the objUser's SID

```
Set objSID = objWMI.Get("Win32_SID.SID="" & objUserName.SID & """)
```

'(2) Create a new blank Win32_Trustee object, and set it's properties to the appropriate

' values

'Create a new blank Win32_Trustee object

```
Set objTrusteeClass = objWMI.Get("Win32_Trustee")  
Set objTrustee = objTrusteeClass.SpawnInstance_()
```

'Set the properties

```
objTrustee.Domain = objSID.ReferencedDomainName  
objTrustee.Name = objSID.AccountName  
objTrustee.SID = objSID.BinaryRepresentation  
objTrustee.SidLength = objSID.SidLength  
objTrustee.SIDString = objSID.SID
```

'(3) Create a new blank Win32_ACE object, and set it's properties to the appropriate

' values. The Trustee property should point to the newly created

Win32_Trustee

'Create a new blank Win32_ACE object

```
Set objACEClass = objWMI.Get("Win32_ACE")  
Set objACE = objACEClass.SpawnInstance_()
```

'Set the properties

```
objACE.AccessMask = PermissionBitMask  
objACE.Trustee = objTrustee  
objACE.AceType = AceType  
objACE.AceFlags = AceFlags
```

microsoft.public.scripting.vbscript: Strange behavior when setting ACL on NTFS Folder

'(4) Add the ACE to the DACL. This is the hard part, because to keep the existing

' ACEs means we have to resize the array. The DACL array is not a dynamic array, so

' the solution is to create a new array of the right size, copy the explicit contents

' of the existing DACL (and the new ACE) into it, and then replace the existing

' DACL with the new one.

'Create a Dictionary object (a hash). We're using a dictionary object, because it

'can be dynamically resized

```
Set objDictionary = CreateObject("Scripting.Dictionary")
```

'Loop through the DACL array and populate the Dictionary object only with ACE objects

'that have EXPLICIT PERMISSIONS. This can be tested by checking the AceFlags property.

'If the 5th bit (16) is set to 1 then, the permission is inherited

(See AceFlags

'description above.

'Doesn't matter what the Dictionary key is, The corresponding item is all that matters,

'being the ACE of the existing DACL. We just use i as the key here

```
For i = LBound(objSD.DACL) to UBound(objSD.DACL)
```

```
  If NOT objSD.DACL(i).AceFlags AND 16 Then
```

```
    objDictionary.Add i, objSD.DACL(i)
```

```
  End If
```

```
Next
```

'Add the new ACE.

```
objDictionary.Add "Empty Key", objACE
```

'Use the items method to return a list of just the items, and overwrite the existing

'DACL with the new list of ACEs

```
objSD.DACL = objDictionary.Items
```

'(5) Finally, write back the Security Descriptor

```
objSD.ControlFlags = 4
```

```
If objSecurity.SetSecurityDescriptor(objSD) = 0 Then
```

```
  SetPermissions = 0
```

```
Else
```

```
  SetPermissions = 1
```

```
  Err.Clear
```

```
End If
```

```
End If
```

microsoft.public.scripting.vbscript: Strange behavior when setting ACL on NTFS Folder

End Function

#####

'END