

Re: System.Security.Principal.WindowsImpersonation

Source: <http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.security/2003-07/0414.html>

From: Shel Blauman [MSFT] (sheldonb_at_online.microsoft.com)

Date: 07/29/03

Date: Tue, 29 Jul 2003 13:39:44 -0700

I'll research this and get back to you.

Shel

--

This posting is provided "AS IS" with no warranties, and confers no rights.

Use of included script samples are subject to the terms specified at

<http://www.microsoft.com/info/copyright.htm>

"borgc" <borgc@polarfab.com> wrote in message

news:3d3d01c355d9\$dac5b4c0\$a001280a@phx.gbl...

> Thanks for the Code, but I have the same problem with
> this code, that I had with my original code. The User is
> impersonated, but the privledges for that user are not
> transfered. If any one has any additional suggestions - I
> would appreciate the help.

>

> Thanks

> Chet

>

> >-----Original Message-----

> >This code should work on XP.

> >

> >Shel

> >

> >[Visual Basic]

> >

> > ' This sample demonstrates the use of the

> > WindowsIdentity class to

> > impersonate a user.

> > ' IMPORTANT NOTES:

> > ' This sample can be run only on Windows XP. The

> > default Windows 2000

> > security policy

> > ' prevents this sample from executing properly, and

> > changing the policy to

> > allow

> > ' proper execution presents a security risk.

> > ' This sample requests the user to enter a password on

> > the console screen.

> > ' Because the console window does not support methods

> > allowing the password

> > to be masked,

> > ' it will be visible to anyone viewing the screen.

Re: System.Security.Principal.WindowsImpersonation

microsoft.public.dotnet.security: Re: System.Security.Principal.WindowsImpersonation

```
> >
> >Imports System
> >Imports System.Runtime.InteropServices
> >Imports System.Security.Principal
> >Imports System.Security.Permissions
> >Imports Microsoft.VisualBasic
> ><Assembly: SecurityPermissionAttribute
> (SecurityAction.RequestMinimum,
> >UnmanagedCode:=True), _
> > Assembly: PermissionSetAttribute
> (SecurityAction.RequestMinimum,
> >Name:="FullTrust")>
> >Module Module1
> >
> >     Public Class ImpersonationDemo
> >
> >         Private Declare Auto Function LogonUser
> Lib "advapi32.dll" (ByVal
> >lpszUsername As [String], _
> >         ByVal lpszDomain As [String], ByVal
> lpszPassword As [String], _
> >         ByVal dwLogonType As Integer, ByVal
> dwLogonProvider As Integer,
> >_
> >         ByRef phToken As IntPtr) As Boolean
> >
> >         <DllImport("kernel32.dll")> _
> >         Public Shared Function FormatMessage(ByVal
> dwFlags As Integer, ByRef
> >lpSource As IntPtr, _
> >         ByVal dwMessageId As Integer, ByVal
> dwLanguageId As Integer,
> >ByRef lpBuffer As [String], _
> >         ByVal nSize As Integer, ByRef Arguments As
> IntPtr) As Integer
> >
> >         End Function
> >
> >         Public Declare Auto Function CloseHandle
> Lib "kernel32.dll" (ByVal
> >handle As IntPtr) As Boolean
> >
> >
> >         Public Declare Auto Function DuplicateToken
> Lib "advapi32.dll"
> >(ByVal ExistingTokenHandle As IntPtr, _
> >         ByVal SECURITY_IMPERSONATION_LEVEL As
> Integer, _
> >         ByRef DuplicateTokenHandle As IntPtr) As
> Boolean
> >
> >         'GetErrorMessage formats and returns an error
> message
> >         'corresponding to the input errorCode.
> >         Public Shared Function GetErrorMessage(ByVal
> errorCode As Integer)
> >As String
> >         Dim FORMAT_MESSAGE_ALLOCATE_BUFFER As
> Integer = &H100
> >         Dim FORMAT_MESSAGE_IGNORE_INSERTS As Integer
> = &H200
> >         Dim FORMAT_MESSAGE_FROM_SYSTEM As Integer =
```

```

> &H1000
> >
> >         Dim messageSize As Integer = 255
> >         Dim lpMsgBuf As String
> >         Dim dwFlags As Integer =
> FORMAT_MESSAGE_ALLOCATE_BUFFER Or
> >FORMAT_MESSAGE_FROM_SYSTEM Or
> FORMAT_MESSAGE_IGNORE_INSERTS
> >
> >         Dim ptrlpSource As IntPtr = IntPtr.Zero
> >         Dim prtArguments As IntPtr = IntPtr.Zero
> >
> >         Dim retVal As Integer = FormatMessage
> (dwFlags, ptrlpSource,
> >errorCode, 0, lpMsgBuf, _
> >         messageSize, prtArguments)
> >         If 0 = retVal Then
> >             Throw New Exception("Failed to format
> message for error code
> >" + errorCode.ToString() + ". ")
> >             End If
> >
> >         Return lpMsgBuf
> >         End Function 'GetErrorMessage
> >         ' Test harness.
> >         ' If you incorporate this code into a DLL, be
> sure to demand
> >FullTrust.
> >         <PermissionSetAttribute(SecurityAction.Demand,
> Name:="FullTrust")> _
> >         Public Overloads Shared Sub Main(ByVal args() As
> String)
> >
> >         Dim tokenHandle As New IntPtr(0)
> >         Dim dupeTokenHandle As New IntPtr(0)
> >         Try
> >
> >
> >         Dim userName, domainName As String
> >
> >         ' Get the user token for the specified
> user, domain, and
> >password using the
> >         ' unmanaged LogonUser method.
> >         ' The local machine name can be used for
> the domain name to
> >impersonate a user on this machine.
> >         Console.WriteLine("Enter the name of a
> domain on which to log
> >on: ")
> >         domainName = Console.ReadLine()
> >
> >         Console.WriteLine("Enter the login of a user
> on {0} that you
> >wish to impersonate: ", domainName)
> >         userName = Console.ReadLine()
> >
> >         Console.WriteLine("Enter the password for
> {0}: ", userName)
> >
> >         Const LOGON32_PROVIDER_DEFAULT As
> Integer = 0

```

microsoft.public.dotnet.security: Re: System.Security.Principal.WindowsImpersonation

```
> >           'This parameter causes LogonUser to
> create a primary token.
> >           Const LOGON32_LOGON_INTERACTIVE As
> Integer = 2
> >           Const SecurityImpersonation As Integer =
> 2
> >
> >           tokenHandle = IntPtr.Zero
> >           dupeTokenHandle = IntPtr.Zero
> >
> >           ' Call LogonUser to obtain a handle to
> an access token.
> >           Dim returnValue As Boolean = LogonUser
> (userName, domainName,
> >Console.ReadLine(), LOGON32_LOGON_INTERACTIVE,
> LOGON32_PROVIDER_DEFAULT,
> >tokenHandle)
> >
> >           Console.WriteLine("LogonUser called.")
> >
> >           If False = returnValue Then
> >               Dim ret As Integer =
> Marshal.GetLastWin32Error()
> >               Console.WriteLine("LogonUser failed
> with error code :
> >{0}", ret)
> >               Console.WriteLine(ControlChars.Cr
> + "Error: [{0}] {1}" +
> >ControlChars.Cr, ret, GetErrorMessage(ret))
> >
> >               Return
> >           End If
> >
> >           Dim success As String
> >           If returnValue Then success = "Yes" Else
> success = "No"
> >           Console.WriteLine(("Did LogonUser
> succeed? " + success))
> >           Console.WriteLine(("Value of Windows NT
> token: " +
> >tokenHandle.ToString()))
> >
> >           ' Check the identity.
> >           Console.WriteLine(("Before
> impersonation: " +
> >WindowsIdentity.GetCurrent().Name))
> >
> >           Dim retVal As Boolean = DuplicateToken
> (tokenHandle,
> >SecurityImpersonation, dupeTokenHandle)
> >           If False = retVal Then
> >               CloseHandle(tokenHandle)
> >               Console.WriteLine("Exception thrown
> in trying to
> >duplicate token.")
> >               Return
> >           End If
> >
> >           ' TThe token that is passed to the
> following constructor
> >must
> >           ' be a primary token in order to use it
```

microsoft.public.dotnet.security: Re: System.Security.Principal.WindowsImpersonation

```
> for impersonation.
> >         Dim newId As New WindowsIdentity
> (dupeTokenHandle)
> >         Dim impersonatedUser As
> WindowsImpersonationContext =
> >newId.Impersonate()
> >
> >         ' Check the identity.
> >         Console.WriteLine("After
> impersonation: " +
> >WindowsIdentity.GetCurrent().Name))
> >
> >         ' Stop impersonating the user.
> >         impersonatedUser.Undo()
> >
> >         ' Check the identity.
> >         Console.WriteLine("After Undo: " +
> >WindowsIdentity.GetCurrent().Name))
> >
> >         ' Free the tokens.
> >         If Not System.IntPtr.op_Equality
> (tokenHandle, IntPtr.Zero)
> >Then
> >             CloseHandle(tokenHandle)
> >             End If
> >         If Not System.IntPtr.op_Equality
> (dupeTokenHandle,
> >IntPtr.Zero) Then
> >             CloseHandle(dupeTokenHandle)
> >             End If
> >         Catch ex As Exception
> >             Console.WriteLine("Exception
> occurred. " + ex.Message))
> >         End Try
> >     End Sub 'Main
> > End Class 'Class1
> >
> >
> >
> >--
> >This posting is provided "AS IS" with no warranties, and
> >confers no rights.
> >Use of included script samples are subject to the terms
> >specified at
> >http://www.microsoft.com/info/copyright.htm
> >
> >
> >"Chet Borg" <borgc@polarfab.com> wrote in message
> >news:02a801c3554c$dd09fa80$a601280a@phx.gbl...
> >> I am writing a application in VB.net which
> >automatically
> >> updates applications on our network. I have everything
> >> working except for registering updated dlls for the vb6
> >> programs on Windows XP workstations (due to the
> >security
> >> settings in Windows XP, non priv users are not allowed
> >to
> >> register dlls) - according to our NT Admin. A colleague
> >> suggested that I use Impersonation, to temporarily
> >> imperonate a priv user.
> >>
> >> When I impersonate the user, the new username is
```

microsoft.public.dotnet.security: Re: System.Security.Principal.WindowsImpersonation

```
> >> returned - however I do not gain the rights associated
> >> with this new user. I was originally using the Win API
> >> DuplicateToken() but switched to the DuplicateTokenEx
> (&),
> >> after some research on the web.
> >>
> >> The code I use for the the function follows.
> >> Any assistance would be appreciated.
> >>
> >>
> >>     Public Function ImpersonateUserAndRegDLLs(ByVal
> >> MyDLLs As c_DLLs) As String
> >>
> >>         Dim sName As String
> >>         Dim token As New IntPtr(0)
> >>         Dim TokenCopy As New IntPtr(0)
> >>         Dim iLogResult As Integer
> >>         Dim newPrincipal As
> >> System.Security.Principal.WindowsPrincipal
> >>         Const SecurityImpersonation As Integer =
> >> System.Management.ImpersonationLevel.Impersonate
> >>         Dim DLL As c_DLL
> >>         Dim TokenAttributes As New SECURITY_ATTRIBUTES
> >>         Try
> >>             'Logs On the User to the Domain
> >>             iLogResult = LogonUser(USERNAME,
> >> Environment.UserDomainName, PWD, LOGON32_LOGON_NETWORK,
> >> LOGON32_PROVIDER_WINNT50, token)
> >>
> >>             'Gets name of current user
> >>             sName =
> >> System.Security.Principal.WindowsIdentity.GetCurrent
> >> ().Name
> >>             'Create a copy of the Token you are trying
> >> to
> >> impersonate
> >>             'Dim retVal As Boolean = DuplicateToken
> >> (token, SecurityImpersonation, TokenCopy)
> >>             Dim iDupResult As Integer =
> DuplicateTokenEx
> >> (token, MAXIMUM_ALLOWED, TokenAttributes,
> >> SECURITY_IMPERSONATION_LEVEL.SecurityImpersonation,
> >> TOKEN_TYPE.TokenPrimary, TokenCopy)
> >>
> >>             sName =
> >> System.Security.Principal.WindowsIdentity.GetCurrent
> >> ().Name
> >>
> >>             'Tests if DuplicateToken works
> >>             If iDupResult = 0 Or iLogResult = 0 Then
> >>                 CloseHandle(token)
> >>                 frm.lstFileHist.Items.Add("Error --
> >> Logging on BuildXP")
> >>             Else
> >>                 'Creates a new Windows Identity
> >>                 Dim NewId As New
> >> System.Security.Principal.WindowsIdentity(TokenCopy)
> >>                 'Performs the impersonation
> >>                 Dim ImpersonatedUser As
> >> System.Security.Principal.WindowsImpersonationContext =
> >> NewId.Impersonate
```

microsoft.public.dotnet.security: Re: System.Security.Principal.WindowsImpersonation

```
> >>
> >>         'Used to Chk impersonation worked
> >>         sName =
> >> System.Security.Principal.WindowsIdentity.GetCurrent
> >> ().Name
> >>
> >>         'Creates a Windows Principal based in
> the
> >> New Identity
> >>         'Used to check if the Users privs were
> >> impersonated also
> >>         newPrincipal = New
> >> System.Security.Principal.WindowsPrincipal(NewId)
> >>
> >>         'Loops through all DLLs and Registers
> Them
> >>         'Regular User is not a member of
> >> Group "Domain Admins", after the impersonation this
> >> should be returned a true
> >>
> >>         MessageBox.Show("The user is a member
> of
> >> Domain Admins: " & newPrincipal.IsInRole("Domain
> >> Admins").ToString)
> >>         MessageBox.Show("User is " & sName)
> >>         MessageBox.Show("Token: " &
> >> System.Security.Principal.WindowsIdentity.GetCurrent
> >> ().Token.ToString)
> >>
> >>         '** impersonation works, but the
> >> impersonated user doesn't get any of the
> >>         '** rights of the user it is
> >> impersonating, so it is unable to register the DLLs.
> >>
> >>         For Each DLL In MyDLLs
> >>             frm.lstFileHist.Items.Add
> ("Attempting
> >> to Register - " & DLL.FileName)
> >>             Dim sResult As String
> >>             'Next Line uses the Shell function
> to
> >> execute a regsvr32 cmd
> >>             sResult = DLL.RegisterDLL()
> >>             If sResult = "TRUE" Then
> >>                 frm.lstFileHist.Items.Add
> >> ("Successfully Executed Regsvr32 for - " &
> DLL.FileName)
> >>             Else
> >>                 frm.lstFileHist.Items.Add
> (sResult)
> >>             End If
> >>         Next
> >>
> >>         'Stop Impersonation
> >> ImpersonatedUser.Undo()
> >>
> >>         'Used to Chk impersonation.undo worked
> >>         sName =
> >> System.Security.Principal.WindowsIdentity.GetCurrent
> >> ().Name
> >>         MessageBox.Show("User is " & sName)
> >>         MessageBox.Show("Token: " &
```

microsoft.public.dotnet.security: Re: System.Security.Principal.WindowsImpersonation

```
> >> System.Security.Principal.WindowsIdentity.GetCurrent
> >> ().Token.ToString)
> >>
> >>         End If
> >>     Catch ex As Exception
> >>         MessageBox.Show(ex.ToString)
> >>     Finally
> >>         If Not System.IntPtr.op_Equality(token,
> >> IntPtr.Zero) Then
> >>             CloseHandle(token)
> >>         End If
> >>         If Not System.IntPtr.op_Equality(TokenCopy,
> >> IntPtr.Zero) Then
> >>             CloseHandle(TokenCopy)
> >>         End If
> >>     End Try
> >> End Function
> >
> >
> >.
```