UNIVERSITÉ ANTONINE Faculté d'ingénieurs en Informatique, Multimédia, Réseaux et Télécommunications



Document 2 – Examples and Implementations

Course: Applications avancées avec C#

Presented by: Elie Matta et al.

Copyright © 2010-2011, eliematta.com. All rights reserved

Table of Contents

Introduction
Example 1 – The DNS4
A.Introduction4
BNET Framework Configuration4
a.Creating a new permission set4
b.Creating a new code group6
c.Exclusive and LevelFinal8
C.Code9
D.Implementation9
Example 2 – Registry 11
A.Introduction11
B.Codes
a.classmain12
b.Button #1
c.Button #212
d.Button #3
C.Implementation13
i.Scenario 113
ii.Scenario 213
iii.Scenario 314
Example 3 – Environnement variables16
A.Introduction16
B.Code
C.Configuration17
D.Implementation19
Example 4 – User Interface
B.Code24
C.Implementation25
Example 5 – Web access27
A.Introduction27

BNET Framework Configuration	27
Modifying permission set	27
C.Code	
D.Implementation	
Example 6 – Printing	
A.Introduction	
BNET Framework Configuration	
Modifying permission set	
C.Code	
D.Implementation	
Example 7 – The active directory	
A.Introduction	
BNET Framework Configuration	
a.Creating a new permission set b.Creating a code group	
C.Code	
D.Implementation	
Note:	

Question 5

Introduction

After reading the security .NET framework security, it is time now to explore all the theory in detailed examples separated as follows:

- 1) The DNS
- 2) Registry
- 3) The environment variables
- 4) User interface
- 5) Web access
- 6) Printing
- 7) The directory service

Two tools shipped with the .NET Framework SDK allow us to configure security: CASpol.exe(Code Access Security Policy Tool) and MSCorCfg.msc (.NET Configuration Tool). The first is a command-line utility, the second a Microsoft Management Console (MMC) snap-in. The graphical user interface of the MSCorCfg MMC snap-in is easier to use and allows you to visualize the overall security configuration more readily. CASpol is quicker and can be used in scripts or batch files. We will use one of the two tools each time we want to configure our implementation. The namespace System.Security.Permissions will be included is in most of our examples.

Please note that we separated this implementation to seven examples to give each example a clearer step by step implementation at each and every example; We could have managed to enclose all of them in one single application with seven buttons testing the seven different permissions.

Example 1 – The DNS Source code path: Examples/Example1/dnstest Using strong name – Declarative security

A. Introduction

Our example consist of demanding simple permission to resolve www.google.com website, we will assign our program to a new permission of our creation and a code group.

We will use a Console application which we will add on it two namespaces: System.Net and System.Security.

B. .NET Framework Configuration

In this example we will use the .NET Framework configuration tool to configure a new permission and a new code group.



a. Creating a new permission set

Expand the **Runtime Security Policy** node. You can see the security policy levels -Enterprise, Machine and User. We are going to change the security settings in Machine policy. First we are going to create our own custom permission set. Right click the **Permission Sets** node and choose **New**. We will name it *NewPermSet*.

NET Framework 2.0 Configuration	
File Action View Help	
ole F Create Permission Set	*
nor	e permissions. Each
uthu m	orization to access a
C Create a new nermission set	the networky and many
Name*	
A NewPermSet	
Description	
u to	o create a new permission set.
ons	s.
C. Import a permission pathrom an VMI file	
B <u>r</u> owse	
	-
< Back Next > Cancel	

In the next figure, we can add permissions to our permission set. In the left panel, we can see all the permissions supported by the .NET Framework. Now get the properties of **DNS** permission. Set "**Grant assemblies unrestricted access to DNS**"



We should add also the **Security permission** to add the permission to execute the code, by selecting security and then checking **Enable assembly execution**.



b. Creating a new code group

Now we will create a code group and set some conditions, so our assembly will be a member of that code group. Notice that in the code groups node, **AII_Code** is the parent node. Right Click the **AII_Code** node and choose **New**. You'll be presented with the **Create Code Group wizard**. We are going to name it *NewCodeGroup*.

			E X
Create a new code	group		
Name:			
NewCodeGroup			
Description			
Description			
12			
-			
C Import a code grou	up from a XML File		
C Import a code grou	up from a XML File	 Brow	se
C Import a code grou	up from a XML File	 Brow	se
C Import a code grou	up from a XML File	Brow	se
C Import a code grou	up from a XML File	 Brow	se
C Import a code grou	up from a XML File	 Brow	se

In the next figure, you have to provide a condition type for the code group. Now these are the **evidence** that we mentioned in Document 1 – page 4. For this example, we are going to use the **Strong Name** condition type. First, sign your assembly (by using sn.exe per example or from the Visual studio by right-clicking on the project name, properties, signing like in the figure below) with a strong name and build the project.

Key file name:			
newdn <mark>s</mark> key			
Protect my key Enter password	file with a password :	N/A	*
Confirm passw	ord:		Calant from Stores
			Select from store
	OK Cance	I III	Select from File
2	OK Cance	1	Select from File Create Test Certificate
eference Paths	OK Cance	More Details	Select from File Create Test Certificate
eference Paths gning	OK Cance	i More Details	Select from File
eference Paths igning ecurity	OK Cance	More Details	Select from File
eference Paths gning ecurity ublish	OK Cance Timestamp server URL: Sign the assembly Choose a strong name key file:	More Details	Select from File

Back to the .NET configuration tool, now press the **Import** button then we select our assembly. Public Key, Name and Version will be extracted from the assembly, so we don't have to worry about them.



Now move on to the next figure. We have to specify a permission set for our code group. Since we have already created one – *NewPermSet*, select it from the list box.

reate	Code Group			X
Wo	uld you liketo use an existing p	ermission set already de	fined in this poli	icy level
orc	reate a new permission set?			
e	Use exisiting permission set:			
	NewPermSet		•	
С	Create a new permission set			

c. Exclusive and LevelFinal

If no one haven't messed around with the default .NET configuration security settings, our assembly should already belong to another built-in code group -

My_Computer_Zone. When permissions are calculated, if a particular assembly falls into more than one code group within the same policy level, the final permissions for that assembly will be the union of all the permissions in those code groups. To calculate permissions you should reference to the Document 1 – page 1, now we need to run our assembly only with our permission set and that is *NewPermSet* associated with the *NewCodeGroup*. So we have to set another property to do just that. Right click the newly created **MyCodeGroup** node and select **Properties**. Check the check box saying "**This policy level will only have the permissions from the permission set associated with this code group.**" This is called the Exclusive attribute. If this is checked then the run time will never grant more permissions than the permissions associated with this code group. The other option is called LevelFinal. These two properties come into action when calculating permissions and they are explained in Document 1 – page 13.

	Membership Condition Permission Set
Codeg	roup name:
NewCor	deGroup
Codeg	roup description:
If the	membership condition is met:
	his policy level will only have the permissions from the
. т	in a poincy level will only indice permissions in one die
₹ P	ermission set associated with this code group
₹ P	ermission set associated with this code group

C. <u>Code</u>

```
//Uses the DnsPermissionAttribute to restrict access only to those who have
permission.
        [DnsPermission(SecurityAction.Demand, Unrestricted = true)]
//Declarative security
        public class MyClass
        {
            public static IPAddress GetIPAddress()
                IPAddress ipAddress =
Dns.Resolve("www.google.com").AddressList[0];
                return ipAddress;
            }
            public static void Main()
            {
                try
                {
                    //Grants Access.
                    Console.WriteLine(" Access granted\n The assigned IP
Address is :" +
MyClass.GetIPAddress().ToString());
                }
                // Denies Access.
                catch (SecurityException securityException)
                {
                    Console.WriteLine("Access denied");
                    Console.WriteLine(securityException.ToString());
                }
```

D. Implementation

It's time to run the code. What we have done so far is, we have put our code into a code group and given unrestricted access to DNS. Run the code it should work fine, resolving correctly the www.google.com website as shown below:

Access granted The assigned IP Address is :66.249.90.104

But if we changed the permission set for our *NewPermSet* setting the DNS permission to **Grant assemblies no access to DNS** in the .NET configuration then we will have an error:

NET Framework 2.0 Configuratio	n	
File Action View Help		
🐤 🔿 🖄 📰 🔏 🛍 🔰		Permission Settings
Create Permission Set		 Grant assemblies <u>n</u>o access to DNS Assemblies without DNS permission will not be able to resolve Internet names or use URL addresses like 'www.microsoft.com'. Grant assemblies <u>unrestricted access to DNS</u>
Available Permissions: Directory Services Event Log Environment Variables File Dialog Isolated Storage File Message Queue Performance Counter Printing Registry Reflection Service Controller Socket Access SQL Client Web Access User Interface X509 Store	Assigned Properties	
	Import	OK <u>Cancel</u>
	< Back	Finish Cancel

When we run the code it will fail to load the DNS because it has no permission



Example 2 – Registry Implementation: Examples/Example2/Implementation Using declarative security – XML configuration file

A. Introduction

A little reminder on the basics of the three levels to get started: "Fully trusted" code can do whatever the user can do (which might be limited if the user is not using a Windows admin account). "No trust" code can't do anything. Anything in between is called "partial trust." Obviously, there are many degrees of partial trust – the code might have almost no privileges at all, or it might be able to do everything except reading the registry or something.

We also remind you that there are three main ways of causing code to run with partial trust:

- 1. If the assembly is loaded from a network share then it will run with the "LocalIntranet" permission set (by default this enforces a number of restrictions such as no registry, no file IO, limited reflection and so on, we will see this in our example).
- 2. If your assembly refuses one or more permissions (per example you declare that you don't want to be able to perform reflection) then your assembly is by definition partially trusted.
- 3. You can configure your ASP.NET application to run with a particular trust level. This is defined in config files; "medium trust" means no registry, no reflection, no file IO outside your app's virtual directory.

Notice that an assembly might be fully trusted or partially trusted depending on the runtime circumstances. The assembly might be fully trusted if loaded from the C: drive, but will be partially trusted if loaded from a network share.

Next, a couple of handy definitions:

Caller – Code that calls other code

Callee – Code that is called by other code

So if method ClientMethod() calls method ServerMethod() then ClientMethod is the caller and ServerMethod is the callee.

If the calling method (the caller) is in an assembly that is running with partial trust then it is a "partially trusted caller." In this case, if the callee's assembly does not have the AllowPartiallyTrustedCallers (APTC) attribute then the called code will not run regardless of the trust level of the callee's assembly. You will get an exception.

B. <u>Codes</u>

The example has two strongly named assemblies: PartialTrustTest.exe and PartialTrustTestLib.dll. There is also a config file called PartialTrustTest.exe.config that is needed for one of the scenarios.

a. <u>classmain</u>

b. Button #1

c. Button #2

```
try
{
    classmain.TestRegistryAccess();
    MessageBox.Show("Registry key was opened", "Partial Trust
Test", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }
    catch (SecurityException exception)
    {
        MessageBox.Show("Can't open registry key: " +
    exception.Message + "; stack = " + exception.StackTrace, "Partial Trust
Test", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
```

d. Button #3

try

```
{
    classmain.AssertRegistryPermissionAndTestAccess();
    MessageBox.Show("Registry key was opened", "Partial Trust
Test", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }
    catch (SecurityException exception)
    {
        MessageBox.Show("Can't open registry key: " +
    exception.Message + "; stack = " + exception.StackTrace, "Partial Trust
Test", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}
```

C. Implementation

i. <u>Scenario 1</u>

Start by placing all the files together in a directory on your C: drive and running the exe. You will see a form with three buttons. They all do the same thing: they try to read the registry. The only difference is in the methods that they use to call the RegistryKey.OpenSubKey method:

- 1. Read the registry (local assembly) calls a method in the EXE
- Read the registry (from PartialTrustTestLib) calls a method in the DLL (see code at bottom of email)
- 3. Read the registry after asserting permission (from PartialTrustTestLib) calls a method in the DLL that asserts the right to read the registry (I'll explain this below, but note that it has nothing to do with the Asserts used in testing)

Click each of the buttons in turn; you should find that they all work fine. This is because both assemblies are fully trusted.

Form1		
Read the registery (local assembly)		
Read the registry (from PartialTrustTestLib)	Partial Trust Test	X
Read the registry after asserting permission (from PartialTrustTestLib)	👔 Registr	y key was opened
		ОК

ii. <u>Scenario 2</u>

Move the DLL to a network drive – let's call that the U: drive and ensure you delete it from the C: drive so that we can be sure which one is being loaded.

Open PartialTrustTest.exe.config in Notepad and uncomment the line. Replace my username (Vista) with yours in the href attribute. Now run the EXE.

Button 1 is still OK, but the other two buttons throw a SecurityException. This is because the DLL is now running with the LocalIntranet permission set and as such it can't read the registry.

The APTC attribute is not relevant here because the caller is fully trusted (it is the callee that is partially trusted).

Form1	
Re	ad the registry (local accembly)
Read the registry a	Form1 Image: Whether the application has occurred in your application. If you click Continue, the application will ignore this error and attempt to continue. If you click Quit, the application will close immediately. Could not load file or assembly 'PartialTrustTestLib, Version=1.0.0.0, Culture=neutral, PublicKeyToken=bd8828e4bf1d441c' or one of its dependencies. The network path was not found.
	▼ Details Continue Quit

Scenario 3 iii.

This is the most interesting one. To set up this scenario, follow these steps:

- 1. Delete PartialTrustTest.exe.config
- 2. Move the EXE to your network drive (U: drive)
- 3. Move the DLL back to your C: drive
- 4. Open the .NET Framework 2.0 Configuration utility and add the DLL to your Global Assembly Cache (GAC) or use the cmd: gacutil.exe /i PartialTrustTestLib.dll
- 5. Now run the EXE

Note that the DLL has the APTC attribute, which it needs here because the exe is no longer running with full trust.

Button 1 fails, because the EXE is now running with the LocalIntranet permission set.

Button 2 fails. Why? The DLL is now fully trusted (it is in the GAC and it doesn't refuse any permissions). But the EXE does not have RegistryPermission. This is an example of tempting (the unprivileged EXE asks the privileged DLL to ask the .NET Framework). To prevent this, the RegistryKey.OpenSubKey method demands that all its callers have RegistryPermission. This demand causes the CLR to do a 'stack walk', checking that each caller in the stack has the required permission.

Button 3 succeeds. This is because it calls a method in the DLL that 'asserts' RegistryPermission.

Notice that the assertion didn't help in scenario 2, because the DLL didn't have RegistryPermission.

Scenario	EXE	DLL	Btn 1	Btn 2	Btn 3
1	C: drive	C: drive	ОК	ОК	ОК
2	C: drive	U: drive	ОК	Fails – callee doesn't have RegistryPermission	Fails – callee doesn't have Registry Permission
3	U: drive	GAC	Fails – callee doesn't have RegistryPermission	Fails – stack walk discovers that EXE doesn't have RegistryPermission	OK – DLL asserts RegistryPermission

To summarise:

Using CAS correctly involves ensuring that your code can't be used maliciously. Our DLL allows partially trusted callers and one of its methods asserts RegistryPermission. Once that DLL is installed in the GAC these two settings lower the security bar considerably. Any assembly that can run on our CLR can load our DLL and use our method to read the registry. Before adding the assertion we should (a) check our method carefully to make sure it can't be used maliciously, and/or (b) apply extra restrictions to our method. The easiest way to restrict the method is to add a Demand to the method. This is where we demand that the callers meet a certain requirement (not the same one that we're asserting, obviously – if they have that then there's no point in asserting it)

Example 3 – Environnement variables Source code path: Examples/Example3/envartest Using XML condition file – Imperative security

A. Introduction

In this example, we will use the CASpol.exe tool to implement our work: The test assembly we'll use is a simple Windows Forms application with a button and a text box. When the user clicks the button, the application demands two arbitrary security permissions, one after the other: a *File IO* permission to read the C:\Windows directory, and an *Environment* permission to read the USERNAME environment variable. If a demand succeeds, a simple message is added to the text box. If either demand fails that is, if the assembly isn't granted the requested permission—the corresponding error string is added to the text box instead:

Environment variables	
Test Permission	

B. <u>Code</u>

And in the button:

```
private void btnTestPerms_Click(object sender, EventArgs e)
{
    try
    {
        FileIOPermission p = new FileIOPermission(
            FileIOPermissionAccess.Read, "C:\\WINNT");
        p.Demand();
        textBox1.Text += "FileIOPermission OK\r\n";
    }
    catch (Exception ex)
    {
}
```

Copyright © 2010-2011, <u>eliematta.com</u>. All rights reserved | **Page** 16

```
textBox1.Text += ex.Message + "\r\n";
}
try
{
    EnvironmentPermission p = new EnvironmentPermission(
        EnvironmentPermissionAccess.Read, "USERNAME");
    p.Demand();
    textBox1.Text += "EnvironmentPermission OK\r\n";
}
catch (Exception ex)
{
    textBox1.Text += ex.Message + "\r\n";
}
```

}

To create a permission set with CASpol, we first need to create an XML file detailing the individual permissions we want (elie.xml):

```
<PermissionSet class="NamedPermissionSet"
  version="1"
  Name="ElieTest"
  Description="Permission set containing my custom permission">
  <IPermission class=
    "System.Security.Permissions.EnvironmentPermission"
    Read="USERNAME"/>
  <IPermission class=
    "System.Security.Permissions.FileIOPermission"
    Read="C:\Windows"/>
  <IPermission class=
    "System.Security.Permissions.SecurityPermission"
    Flags="Execution"/>
  <IPermission class=
    "System.Security.Permissions.UIPermission"
    Unrestricted="true"/>
</PermissionSet>
```

The *<PermissionSet>* root element must be present and has a class attribute with the value *NamedPermissionSet* or *System.Security.NamedPermissionSet*. For this version of the .NET Framework, the version attribute is *1*. The *Name* attribute is the name of the permission set as it appears in the MSCorCfg tree list, and the *Description* attribute is any arbitrary description of the permission set, which appears in the MSCorCfg right-hand pane. The *<PermissionSet>* element can contain any number of *<IPermission>* elements, which represent the permissions in the permission set—these can be Framework library classes or custom permission classes.

C. Configuration

Now we can start applying the XML as a first step to the security policy in one of two ways. The first approach uses MSCorCfg: select the Permission Sets node for the policy level you want (in this case Machine), and click the Create New Permission Set link. Select the **Import a permission set from an XML file** option and we specified the path to the XML file containing the permissions, as shown in this figure:

NET Framework 2.0 Configuration	
File Action View Help	
🗢 🔿 🙍 🗔 📋 🛛 🖬	Create Permission Set
 Assembly Cache Configured Assemblies Remoting Services Runtime Security Policy Enterprise Machine Code Groups Code Groups Code Groups Code Groups All_Code My_Computer_Z(Code Internet_Zone Internet_Zone FullTrust SkipVerification Execution Nothing LocalIntranet Internet Internet Internet Internet Internet Internet Internet Everything Policy Assemblies 	Pe Oreate a newpermission set Name: Description: Description: Oreate a permission set from an XML file. C:\Users\Vista\Desktop\elie.xml Browse
	< Back Finish Cancel

Using CASpol instead, you use this command line:

caspol -machine -addpset elie.xml

where *-machine* indicates the policy level to add the permission set to, and elie.xml is the file containing the required permissions. CASpol will prompt to make sure you want to proceed—enter **Y** to confirm. Whichever way you've set up these permissions, you can now examine them in MSCorCfg. (You might need to close and reopen MSCorCfg before the changes are displayed.) You can also list the permissions with this CASpol command line:

caspol -machine -listpset

This command will produce a listing of all permission sets for the specified policy level. Somewhere in the middle of this list we should recognize your custom permission set:

8. ElieTest (Permission set containing my custom permission) =
 <PermissionSet class="System.Security.NamedPermissionSet" version="1"
 Name="ElieTest" Description="Permission set containing my custom permission">

<IPermission class="System.Security.Permissions.EnvironmentPermission, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Read="USERNAME" />

<IPermission class="System.Security.Permissions.FileIOPermission, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Read="C:\Windows" />

<IPermission class="System.Security.Permissions.SecurityPermission, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Flags="Execution" />

<IPermission class="System.Security.Permissions.UIPermission, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Unrestricted="true" /> </PermissionSet>

The second operation is to add a new code group, including its membership condition and permission set. In the following command, All_Code is the parent code group to which we want to add a new child, the membership condition is a specified file URL, the permission set is *ElieTest*, the new code group will be named Elie_Group, and we want this to apply exclusively:

caspol -addgroup All_Code -url file://C:/environmentpermtest/* ElieTest -n Elie_Group -exclusive on

c:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE>caspol -addgroup All_Code -url file://C:/environmentpermtest/* ElieTest -n Elie_Group -exclusive Microsoft (R) .NET Framework CasPol 2.0.50727.4016 Copyright (c) Microsoft Corporation. All rights reserved.

The operation you are performing will alter security policy. Are you sure you want to perform this operation? (yes/no)

Added union code group with "-url" membership condition to the Machine level. Success

We created the C:\environmentpermtest directory to test it in our implementation.

D. Implementation

Build and test the application as is before making any coding or security configuration changes



Now to test the application. From Windows Explorer, copy the envartest.exe assembly into the newly configured C:\environmentpermtest directory and double-click it to run it from there. At this stage, everything should run as before, with all permission requests granted. For the second test, in the MSCorCfg snap-in, select the Elie_Group code group and click the Edit Code Group Properties link. From the dialog box, select the Permission Set tab and change the permission set for this code group to, say, LocalIntranet. Then click OK.



We'll obtain the following when we try to run envartest.exe from C:\environmentpermtest



Now try to run the application again: you should find that although the application still executes and the request for the specific Environment permission succeeds, the request for File IO permission fails. Change the permission set again, this time to Internet. This time, although the application executes, the runtime presents an alert message (shown in the figure below) indicating that the assembly is running in a partially trusted security context and warning you that some functionality might not be available.

You can click the alert message to remove it. Indeed, when you click the Test button, you'll find that neither of the requested permissions has been granted.

	NyComputer - Environment v 🗆 💷 💌
MyComputer - Environment v 🗖 🗖 💌	Test Permission
Microsoft .NET Security Warning Never enter personal information or passwords into a window unless you can verify and trust the source of the request. Source: VISTA	Request for the permission of type 'System.Security.Permissions.FileIOPermission, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089' failed. Request for the permission of type 'System.Security.Permissions.EnvironmentPermis sion, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089' failed.

If we make one final test and change the permission set to *Nothing*, we'll find that the application won't even execute and the runtime will throw an exception. To round out your understanding of the various predefined permission sets (as well as any custom sets we've set up), we could experiment with replacing the permission requests as indicated in the following code, first to *AllAccess* for the root of C:\, and then to *PermissionState.Unrestricted*:

```
FileIOPermission p = new FileIOPermission(
//FileIOPermissionAccess.Read, "C:\\Windows");
//FileIOPermissionAccess.AllAccess, "C:\\");
PermissionState.Unrestricted);
```

Example 4 – User Interface Source code path: Examples/Example4/uipermset Using URL – Imperative security

A. Introduction

In this example we will test the UIPermission with both FileIOPermission and RegistryPermission to test respectively the behavior of our window, the access to our local drives and to our registry.

We should first define the type of windows that our code is allowed to use, defined in this table:

Member name	Description
AllWindows	Users can use all windows and user input events without restriction.
NoWindows	Users cannot use any windows or user interface events. No user interface can be used.
SafeSubWindows	Users can only use SafeSubWindows for drawing, and can only use user input events for user interface within that subwindow. Examples of SafeSubWindows are a MessageBox, common dialog controls, and a control displayed within a browser.
SafeTopLevelWindows	 Users can only use SafeTopLevelWindows and SafeSubWindows for drawing, and can only use user input events for the user interface within those top-level windows and subwindows. When it runs under SafeTopLevelWindows permission, your application: Will show the DNS name or IP address of the Web site from which the application was loaded in its title bar. Will display Balloon tool-tip when it first displays, informing the user that it is running under a restricted trust level. Must display its title bar at all times. Must display window controls on its forms. Cannot minimize its main window on startup. Cannot use the Opacity property on Form to make its windows less than 50% transparent.

 Must use only rectangular windows, and must include the window frame. Windows Forms will not honor setting FormBorderStyle to None. Cannot make windows invisible. Any attempt by the application to set the Visible property on its Form objects to False will be ignored. Must have an entry in the Task Bar. Will have its controls prohibited from accessing the Parent property. By implication, controls will also be barred from accessing siblings - i.e., other controls at the same level of nesting. Cannot control focus using the Focus method. Will have restricted keyboard input access, so that a form or control can only access keyboard events for itself and its children.
 Will have restricted mouse coordinate access, so that a form or control can only read mouse coordinates if the mouse is over its visible area. Cannot set the TopMost property. Cannot control the z-order of controls on the form using the BringToFront and SendToBack methods. These restrictions help prevent potentially harmful code from spoofing attacks, such as imitating trusted system dialogs.

B. <u>Code</u>

```
class MainClass
{
    public static void Main()
    {
        RegistryPermission f = new
RegistryPermission(RegistryPermissionAccess.Read,
"HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0");

    if (f.IsUnrestricted())
        Console.WriteLine("Unrestricted Access allowed");
        else
        Console.WriteLine("Unrestricted Access DENIED");
```

```
FileIOPermission fileIO = new
FileIOPermission (PermissionState.None);
        Console.WriteLine("All Local files read access: {0}",
FileIOPermissionAccess.Read);
        Console.WriteLine("All Local files write access: {0}",
FileIOPermissionAccess.Write);

        UIPermission(UIPermissionWindow.AllWindows,
UIPermissionClipboard.AllClipboard);
        if (ui.IsUnrestricted())
            Console.WriteLine("UI Unrestricted Access allowed");
        else
            Console.WriteLine("UI Unrestricted Access DENIED");
    }
}
```

C. Implementation

After running the code in a console application, we will have a result like this:



That's because we are running the code while grating access to see and use the window normally but we don't have permission to the required registry path. If we simply change the UIPermissionWindow to NoWindows then we'll obtain:

Unvestwicted Access DENIED
All Local files read access: Read
HII LOCAL FILES WRITE ACCESS: WRITE
UI Upwootwistad Assass DENIED
OI UNPESTFICTEU ACCESS DEMIED
Ducco any key to continue
rress any key to continue

Now if we put the uipermtest.exe in the C:\environmentpermtest while changing the Elie_Group permission to Nothing:

Reg. NET Framework 2.0 Configuration	
File Action View Help	
♦ ♦ 2 📰 4 🖻 1 🗶 1 12 5	
Isole Roo Elie_Group Properties	
NET Frat	
Assemblies that meet this code group's membership condition will receive the permissions in the permission set specified below.	m permission
Ri Permission set:	
A 💀 Rine ElieTest	
▷ 액 FullTrust SkipVerification Execution Nothing	ons in this permission set and to view and It-clicking them in the permission list.
LocalIntranet Internet ElieTest Everything	Illows you to add or remove permissions d also allows you to modify permissions in rmission and clicking Properties .
	ermission set another name. Be careful code groups depending on the old name
View Permission	gle between this Help topic and the
OK Cancel Apply	
Done	

The code will fail to start and an exception will be raised:

C:\>cd C:\environmentpermtest C:\environmentpermtest>uipermtest.exe Unhandled Exception: System.IO.FileLoadException: Could not load file or assembly 'uipermtest, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null' or sion to execute. (Exception from HRESULT: 0x80131418) File name: 'uipermtest, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null' ---> System.Security.Policy.PolicyException: Execution permission cannot at System.Security.SecurityManager.ResolvePolicy(Evidence evidence, PermissionSet reqdPset, PermissionSet optPset, PermissionSet denyPset, Permission at System.Security.SecurityManager.ResolvePolicy(Evidence evidence, PermissionSet reqdPset, PermissionSet optPset, PermissionSet denyPset, Permission at System.Security.SecurityManager.ResolvePolicy(Evidence evidence, PermissionSet reqdPset, PermissionSet optPset, PermissionSet denyPset, Permission at System.Security.SecurityManager.ResolvePolicy(Evidence evidence, PermissionSet reqdPset, PermissionSet optPset, PermissionSet denyPset, Permission at System.Security.SecurityManager.ResolvePolicy(Evidence evidence, PermissionSet reqdPset, PermissionSet optPset, PermissionSet denyPset, Permission

Example 5 – Web access Source code path: Examples/Example5/webacctest Using strong name – Imperative security

A. Introduction

Our example consist of connecting and accepting webpermissions to google.com, msn.com and yahoo.com.

We will use regular expressions, also referred to as regex to provide a concise and flexible means for matching strings of text in this case on www.google.com, and IEnumerator for a simple iteration over a collection.

We will use a Console application adding the System.Text.RegularExpressions, System.Net and System.Collections namespaces.

B. .NET Framework Configuration

In this part we will use the permission set *NewPermSet* and code group *NewCodeGroup* already used in Example 1.

Modifying permission set

Expand the **Runtime Security Policy** node. You can see the security policy levels -Enterprise, Machine and User. We are going to change the security settings in Machine policy. We will add the Security and Web access and grant it unrestricted access because we will control it in the code.



Security:



For this example, we are going to use the **Strong Name** condition type. First, sign your assembly (by using sn.exe per example or from the Visual studio by right-clicking on the project name, properties, signing) with a strong name and build the project.

Create Strong Name Key
Key file name:
newwebtest
Protect my key file with a password
Enter password:
Confirm password:
OK Cancel

Back to the .NET configuration tool, now press the **Import** button and select your assembly. Public Key, Name and Version will be extracted from the assembly.

Choose the o	condition type for this code group:	
Strong Name		
strong name this member with this cod Provide the s optional but	ame memoersnip conditions true for all assemblies with a that matches the one defined below. Assemblies that meet ship conditionwill be granted the permissions associated e group. strong name's public key. The name and version are provide a more secure condition.	
Public key:	002400000480000094000000602000002400005253413	
Name:	webacctest	
Version:	1.0.0.0	

Now move on to the next figure. We have to specify a permission set for our code group. Since we have already created one – *NewPermSet*, select it from the list box.

Would you like to use an existing permissions	set already de	fined in this polic	y level
or create a new permission set?			
• Use exisiting permission set:			
NewPermSet		•	
C Create a new permission set			
	< Back	Next >	Cancel

Now we will go to .NET configuration and set the option "This policy level will only have the permissions from the permission set associated with this code group"

eneral	Membership Condition Permiss	sion Set
Codeg	roup name:	
NewCo	deGroup	
Code g	roup description:	
If the	membershin condition is met:	
If the	membership condition is met: his policy level will only have the	permissionsfrom the
If the	membership condition is met: his policy level will only have the ermission set associated with th	permissionsfrom the ris code group
If the T p P	membership condition is met: his policy level will only have the ermission set associated with th olicy levels below this level will	permissionsfrom the nis code group not be evaluated

C. Code

```
// Create a Regex that accepts all URLs containing the host fragment
www.google.com.
            Regex myRegex = new Regex(@"http://www\.google\.com/.*");
            // Create a WebPermission that gives permissions to all the hosts
containing the same host fragment.
            WebPermission myWebPermission = new
WebPermission (NetworkAccess.Connect, myRegex);
            //Add connect privileges for a www.msn.com.
            myWebPermission.AddPermission(NetworkAccess.Connect,
"http://www.msn.com");
            //Add accept privileges for www.yahoo.com.
            myWebPermission.AddPermission(NetworkAccess.Accept,
"http://www.yahoo.com/");
            // Check whether all callers higher in the call stack have been
granted the permission.
            myWebPermission.Demand();
            // Get all the URIs with Connect permission.
            IEnumerator myConnectEnum = myWebPermission.ConnectList;
            Console.WriteLine("\nThe 'URIs' with 'Connect' permission are
:\n");
            while (myConnectEnum.MoveNext())
            { Console.WriteLine("\t" + myConnectEnum.Current); }
            // Get all the URIs with Accept permission.
            IEnumerator myAcceptEnum = myWebPermission.AcceptList;
            Console.WriteLine("\n\nThe 'URIs' with 'Accept' permission is
:\n");
```

Copyright © 2010-2011, <u>eliematta.com</u>. All rights reserved | **Page**

```
while (myAcceptEnum.MoveNext())
{ Console.WriteLine("\t" + myAcceptEnum.Current); }
```

D. Implementation

It's time to run the code. What we have done so far is, we have put our code into a code group and given unrestricted access to Web Access. Run the code it should work fine, giving access to www.google.com/* which is everything that follows www.google.com and accepts connection from www.yahoo.com:



Example 6 – Printing Source code path: Examples/Example6/printtest Using strong name – Declarative security

A. Introduction

Our example consist of demanding printing permission to print a document by a default printer, we will also use permission flags like RequestMinimum explained in Document 1 – page 5

We will use a Windows Form Application which we will add on it two namespaces: System.Drawing.Printing and System.Security.Permissions.

First there are essential types of access that we can allow to our code which are:

- AllPrinting Provides full access to all printers.
- **DefaultPrinting** Provides printing programmatically to the default printer, along with safe printing through showing a less restricted dialog box. DefaultPrinting is a subset(a subset is a set contained with another set) of AllPrinting.
- NoPrinting Prevents access to printers. NoPrinting is a subset of SafePrinting.
- **SafePrinting** Provides printing only from showing a restricted dialog box. SafePrinting is a subset of DefaultPrinting.

B. .NET Framework Configuration

In this part we will use the permission set *NewPermSet* and code group *NewCodeGroup* already used in Example 1.

Modifying permission set

Expand the **Runtime Security Policy** node. You can see the security policy levels -Enterprise, Machine and User. We are going to change the security settings in Machine policy. We will add the Security, User Interface and grant it unrestricted access, we will also add Printing with permission: Default Printing :



For this example, we are going to use the **Strong Name** condition type. This time we will sign the assembly by using cmd (Command Prompt) by typing the following command in the correct path:

sn.exe -- k "path\nameofthekey.snk"

In our case:

Administrator: C:\Windows\system52\cmu.exe

sn.exe -k "E:\Lilo\UPA\Quatrieme annee\Premier semestre (S7)\Application avancee avec C#\C# project\Document\Examples\example6\newprintkey.snk"

Setting environment for using Microsoft Visual Studio 2008 x86 tools.

c:\Program Files\Microsoft Uisual Studio 9.0\Common7\IDE)sn.exe -k "E:\Lilo\UPA\Quatrieme annee\Premier semestre (S7)\Application avancee avec C#\C# project\

Microsoft (R) .NET Framework Strong Name Utility Version 3.5.21022.8 Copyright (c) Microsoft Corporation. All rights reserved.

Key pair written to E:\Lilo\UPA\Quatrieme annee\Premier semestre (S7)\Application avancee avec C#\C# project\Document\Examples\example6\newprintkey.snk

A key is generated, but it is not yet associated with the assembly of the project. To create this association, double-click the AssemblyInfo.cs file in Visual Studio .NET Solution Explorer. This file has the list of assembly attributes that are included by default when a project is created in Visual Studio .NET. Modify the **AssemblyKeyFile** assembly attribute in the code as follows:

```
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile("E:\\Lilo\\UPA\\Quatrieme annee\\Premier semestre
(S7)\\Application avancee avec C#\\C#
project\\Document\\Examples\\example6\\newprintkey.snk")]
[assembly: AssemblyKeyName("")]
```

Compile the project by clicking CTRL+SHIFT+B. You do not have to have any additional code to install a .dll file in the GAC.

Back to the .NET configuration tool, now press the **Import** button and select your assembly. Public Key, Name and Version will be extracted from the assembly.

Create Code Gr	oup			×	
Choose	the condition type for t	iis code group:	1	-	nu to configuro accombliac
Strong	ame		-	- L	cy policy specific to version
vith this Provide optiona Public k	code group. the strong name's publ but provide a more sec 29: 00240000048000	c key. The name and ve ure condition.	rsion are	3	Configuration tool will not is of the .NET Framework. adjust settings for those
✓ Name	: printtest			-	
🔽 Versi	on: 1.0.0.0			-	
Use the from an	import button to retriev assembly.	e the strong name	Import		re designed to be shared by to view, add, and remove the s computer.
					ies from the assembly cache that an determine which version of
		< Back	Next >	Cancel	ed to load the assembly.

Now move on to the next figure. We have to specify a permission set for our code group. Since we have already created one – *NewPermSet*, select it from the list box.

Create Code Group			×
Would you like to use an existing permiss	ion set already de	fined in this poli	cy level
or create a new permission set?			
 Use exisiting permission set: 			
NewPermSet		•	
C Create a newpermission set			

Now we will go to .NET configuration and set the option "This policy level will only have the permissions from the permission set associated with this code group"

	Membership Condition Permission Set	
Codeg	roup name:	
NewCoo	deGroup	
Code g	roup description:	
	membership condition is met:	
If the		
If the	his policy level will only have the permissions from the ermission set associated with this code group	
If the P	This policy level will only have the permissions from the ermission set associated with this code group 'olicy levels below this level will not be evaluated	

C. <u>Code</u>

In the AssemblyInfo.cs:

```
[assembly: PrintingPermission(SecurityAction.RequestMinimum,Level =
PrintingPermissionLevel.DefaultPrinting)]
```

Button #1:

```
try
{
    PrintDocument mydoc = new PrintDocument();
    mydoc.Print();
    MessageBox.Show("This code can print");
    }
    catch (Exception ex)
    {
        MessageBox.Show("This code cannot print because "
        +ex.Message);
    }
}
```

D. Implementation

It's time to run the code. What we have done so far is, we have put our code into a code group and given unrestricted access to User Interface and Security. Run the code it should work fine, printing a document by the default printer, in our case the default printer is to send the document to Microsoft Office OneNote:

🕼 Untit	led page - Microsoft Office OneNote	
<u> </u>	Printing Per D Z E Tools Table Window Help A	
- 🕲	<table-of-contents> 🚵 🔊 • 🕲 • 🕅 Task • 🚖 Tag • 🚭 Clip 🖽 🛷 🎢 • 🥹 100%</table-of-contents>	• 🗸 17 • B 🗄 • 🎒 •
»	Test Printing	Search All Notebooks 🔎 🔻
Work Notebool	Friday, December 18, 2009	New Page
Personal Noteb	3:11 PM This code can print OK	-
OneNote		
	<	► -

On the other hand, if we go back to .NET configuration tool and change the printing permission to No Printing, we will have an error as soon as we try to runt the code because we don't have the minimum permission to run the code and to print.

Create Permission Set		
	Permission Settings	×
Available Permissions: Directory Services Directory Services Directory Services Directory Services Directory Service File Dalog File Dalog Properties Isolated Storage File Message Queue Performance Counter Registry Reflection Service Controller Sold Lacess SQL Client Web Access X509 Store	Assigned Permissions: Assigned Permissions: Security User Interface Protocy	vinters: T
Import	C Grant assemblies unrestricted access to p	rinters
		OK Cancel

The error:



Example 7 – The active directory Source code path: Examples/Example7/adtest Using publisher – Imperative security

A. Introduction

Our example consist of connecting to the active directory on our computer and listing all the users that belongs to the administrators group.

We will use IEnumerator for a simple iteration over a collection.

We will also have to add the System.DirectoryServices namespace manually if it doesn't exist, as shown in the figure below:

Soluti	on Exp	olorer - adtest 🛛 👻 🖛 🗙	2 💘	Add Reference	? <mark>x</mark>
	P	5	Sole	-	
	Solutio	on 'adtest' (1 project)	tion	.NET COM Projects Browse Recent	
		Build			
E E		Rebuild		Component Name	Version 🔦
Ē		Clean		System.Data.Services.Client	3.5.0.0
		Publish		System.Data.Services.Design	3.5.0.0
		Add	•	System.Data.SqlXml	2.0.0.0
		Add Reference		System.Deployment	2.0.0.0
		Add Web Reference		System.Design	2.0.0.0
	æ,	View Class Diagram		System.DirectoryServices	2.0.0.0
		Set as StartUp Project		System.DirectoryServices.AccountManagement	3.5.0.0
		Debug	•	System.DirectoryServices.Protocols	2.0.0.0
	¥	Cut		System.Drawing	2.0.0.0
		Paste		System.Drawing.Design	2.0.0.0
	×	Remove		System.EnterpriseServices	2.0.0.0 👻
		Rename		<	4
		Unload Project	- II I		
	ß	Open Folder in Windows Explorer		ОК	Cancel
		Properties			

B. .NET Framework Configuration

In this part we will use a new code group *ADCodeGroup* and a new permission set *ADPermSet*.

a. Creating a new permission set

We will add the Security, User Interface and Environment variables and grant them unrestricted access to be able to bring the names of the administrators, and we will add the Directory Services with path "**WinNT**://**Vista,Computer**" to browse through the active directory as shown in the next figure:

Create Permission Set	WT harmont 21 and go and 1 the	Permission Settings Grant assemblies access to the follow	wing directory service paths:
Available Permissions: DNS Event Log File IO File Dialog Isolated Storage File Message Queue Performance Counter Printing Registry Reflection Service Controller Socket Access SQL Client Web Access Store	Assigned Permissions: Add >> Environment Variables Security User Interface	Path WinNT://Vista,Computer	Access Browse Delete Entry
Directory Services	Import	C Grant assemblies unrestricted acce	ss to all directory service paths

For this example, we are going to sign the certificate and import it to the **publisher** in .NET framework. First, we will sign our assembly by the ClickOnce manifests (from the Visual studio by right-clicking on the project name, properties, signing) by creating a test certificate.

Build		
Build Events	Image: Sign the ClickOnce manifests	
Debug	Certificate:	
Resources	Issued To (none) Issued By (none)	Select from Store
Services	Intended Purpose (none) Expiration Date (none)	Select from File
Settings		Create Test Certificate
Reference Paths	More Details	
Signing*	Timestamp server URL:	
Security		
Publish	Sign the assembly	
		Change Password
	Delay sign only	

Then we will have to enter a password, we've put: mynewpass

Enter password for new file adte	st_TemporaryKey.pfx
•••••	
Confirm your password	

Note that as from now on the certificate has been created as we can notice the (none) fields has been replaced by the newly created certificate:

Issued To	VISTA\Vista	Calant forms Chang
Issued By	VISTA\Vista	Select from Store
Intended Purpose	<all></all>	
Expiration Date	12/27/2010 7:09:16 PM	Select from File
		Create Test Certificate

Then, click on the More Details button and then click on the install certificate:

tificate	Certificate Import Wizard
Details Certification Path Image: Certificate Information Image: Certificate Information This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.	Welcome to the Certificate Import Wizard This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store. A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept. To continue, dick Next.
Issued to: VISTA\Vista	
Issued by: VISTA\Vista	
Valid from 12/27/2009 to 12/27/2010 You have a private key that corresponds to this certificate.	
Install Certificate Issuer Statement Learn more about certificates Issuer Statement	< Back Next > Cancel
OK	

Click on next, then you should choose the Place all certificate in the following store and click Browse then select "**Trusted Root Certification Authorities**" to sign properly this certificate to a high level. Finally click on finish, you might have this security warning that tells you that a newly certificate has been added to the root certification list and it is not trusted by Microsoft:

Security Wa	arning	23
	You are about to install a certificate from a certification authority (CA) claiming to represent: VISTA\Vista Windows cannot validate that the certificate is actually from "VISTA\Vista". You should confirm its origin by contacting "VISTA\Vista". The following number will assist you in this process:	
	Thumbprint (sha1): 17CD10A6 92C07504 AEEF1DCB 69264A29 D5F0153B	
	Warning: If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.	
	Do you want to install this certificate?	
	<u>Y</u> es <u>N</u> o	

Click on yes, and the import should be successful.

b. Creating a new code group

Back to the .NET configuration tool, we will choose the **Publisher** for the condition type, then click on Import from Signed file and choose the name of your project with an extension: **.vshost.exe** as shown in the figure below:

Code Group	an 177 / 1	23	Import Publisher Certificate F	rom Signed File		
242			🔾 🖓 😺 🕷 adtest)	bin 🕨 Debug	✓ 49 Search	م
			🎍 Organize 👻 🖪 Views	👻 📑 New Folder		2
Choose the condition typ	pe for this code group:		Favorite Links	Name Date	modified Type	Size
Publisher	-		🧮 Desktop			
The Publisher membershi signed with a certificate ti meet this membership co associated with this code Publisher certificate deta	ip condition is true if an assembly is digitally hat matches the one below. Assemblies that indition will be granted the permissions a group. ills:		Computer Co	adtest.exe	adtest.vshost.exe	
Property	Value		Public			
lame						
Issuer Name						
Hash						
Import from Signed	File Import from Certificate File					
			Folders ^			
	< Back Next >	Cancel	File name	adtest.vshost.exe	✓ Assemblies	s (*.dll, *.exe) 🔹
					Open	Cancel

The import should be successful and all the fields should be filled automatically:

The Publisher membership	condition is true if an assembly is digitally
ngine of the second sec	it matches the one below. Assemblies that dition will be granted the permissions roup.
Property	Value
Name	CN=Microsoft Corporation, O=Micr
Issuer Name	CN=Microsoft Code Signing PCA, O
Hash	D57FAC60F1A8D34877AEB350E83F

Click on Next, and choose ADPermSet.

Create Code Group	X
	-1
or create a new permission set?	ei
 Use exisiting permission set: 	
ADPermSet	
Create a new permission set	
	Cancal
< Dalk Next >	Cancer

Now we will go to .NET configuration and set the option "This policy level will only have the permissions from the permission set associated with this code group" ADCodeGroup Properties

ADCode	eGroup	
Codeg	roup description:	
If the	membership condition is m	et:
If the	membership condition is m his policy level will only h	et: avethe permissions from the
If the	membership condition is m his policy level will only h ermission set associated	et: avethe permissions from the I with this code group

C. <u>Code</u>

```
try
            {
                DirectoryServicesPermission dsp = new
DirectoryServicesPermission(DirectoryServicesPermissionAccess.Browse,
"WinNT://" + Environment.MachineName + ",Computer");
                DirectoryEntry localmachine = new DirectoryEntry("WinNT://" +
Environment.MachineName + ",Computer");
                DirectoryEntry admgroup =
localmachine.Children.Find("administrators", "group");
                object members = admgroup.Invoke("members", null);
                foreach (object groupmember in (IEnumerable)members)
                {
                    DirectoryEntry member = new DirectoryEntry (groupmember);
                    textBox1.Text += "" + member.Name + "\r\n";
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
```

D. Implementation

It's time to run the code. What we have done so far is, we have put our code into a code group and given unrestricted access to Security and User Interface and to Environment variables, and to browse into the Directory Services. Run the code it should work fine like shown in the next figure:



If we tried to modify the code to write into the active directory:

```
DirectoryServicesPermission dsp = new
DirectoryServicesPermission(DirectoryServicesPermissionAccess.Write,
"WinNT://" + Environment.MachineName + ",Computer");
```

Then we will have eventually an error:

Microsof	ft Visual Studio				
	An unhandled exception of type 'System.Security.SecurityException' occurred in System.Windows.Forms.dll				
	Additional information: Request for the permission of type 'System.DirectoryServices.DirectoryServicesPermission, System.DirectoryServices, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' failed.				
	Break Continue Ignore	•			

Note:

You can at any time evaluate the assembly to check at which permission or code group the selected assembly have, just follow the steps as shown in the figures below:



This is the evaluation of the adtest assembly:

