BONUS ARTICLE!

Implement Security Policies in .NET

Learn how to grant assemblies permission to perform operations safely with external entities.

by Juval Löwy

Technology Toolbox

- ☑ VB.NET
- ☑ C#
- ❑ SQL Server 2000
- ❑ ASP.NET
- ❑ XML
- ❑ VB6
- ☑ Other:
  .NET Framework

Component-oriented security was difficult to achieve in the unmanaged code world. Today you can achieve component-oriented security in .NET by granting assemblies certain permissions to perform operations with external entities such as the file system, the Registry, the user interfaces, and so on. I'll describe the core concepts and default policies of component-oriented security and show you how to define custom permission sets suitable to your needs.

A security permission is an individual grant to perform a specific operation. When an assembly tries to perform any protected operation or access a resource at run time, .NET verifies that the assembly and its calling assemblies have the appropriate security permissions. .NET provides multiple ways to identify which assembly to grant what permission, and what evidence the assembly

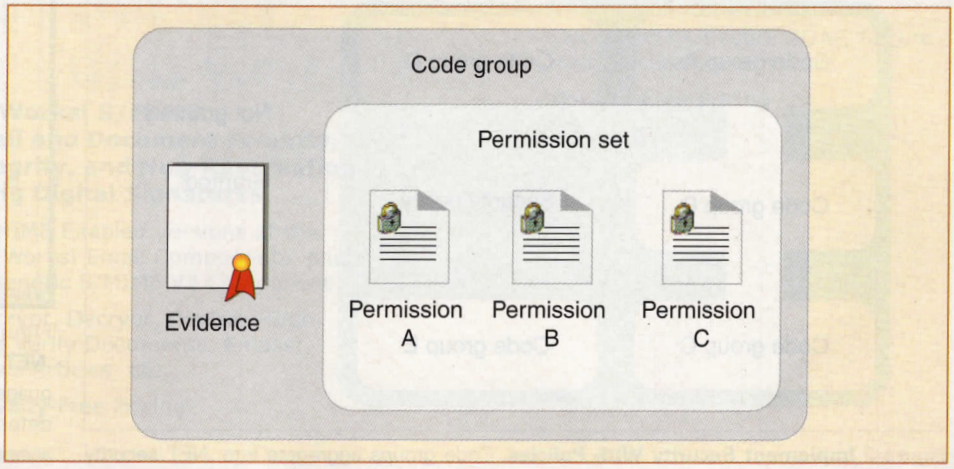

Figure 1 Bind Evidence to a Permission Set. .NET security is based on granting assemblies permissions to perform operations on the file system, the Registry, the user interfaces, and other external entities. You use code groups to classify assemblies so .NET can determine which security permissions to grant. A code group binds a single permission set to a single piece of evidence. An assembly is granted permission in the permission set when the assembly identity matches the evidence.

Go Online!

Use these Locator+ codes at www.visualstudiomagazine.com to go directly to these related resources.

Discuss

VS0301JL_D Discuss this article in the .NET Framework forum.

Read More

VS0301JL_T Read this article online. It includes Table 1 and two extra figures.

NM0205JL_T "Reduce Application Maintenance Costs" by Juval Löwy

should provide to establish its identity.

Security permissions have both type and scope. A file I/O permission's type differs from that of a user interface permission, because each controls access to different types of resources. A permission's scope can be narrow, wide, or unrestricted. For example, a file I/O permission can allow reading from a particular file. You can represent writing to the same file with different file I/O permissions. .NET defines 19 types of permission. These govern the operations and resourses an application is likely to use, such as file I/O, UI, Web access, database access, network, reflection, and other types, including special security permissions.

You often need a set of permissions of various scopes and types for an assembly to function properly. You can create collections of individual permissions called permission sets. You can also use existing sets called named permission sets. .NET provides seven named permission sets: Nothing, Execution, Internet, LocalIntranet, Everything, FullTrust, and SkipVerification (download Table 1 from the *VSM* Web site; see the Go Online box for details).

.NET grants permissions to assemblies based on their identities. Security evidence is some form of proof an assembly can provide to validate its identity. Assemblies substantiate their identities with origin- and assembly-based security evidence.

Origin-based evidence simply examines where the assembly is coming from, independent of content. Standard origin-based evidence consists of code coming from the application directory, a particular Web site, a URL, or a zone. A zone is the local machine, an intranet, the Internet, or an explicitly trusted or untrusted Internet site.

Assembly-based evidence examines assembly content. It looks for a specific match with specified criteria. Standard assembly-based evidence consists of a specific strong name, publisher certificate, assembly hash, and All Code—a wild card satisfied by all code.

You use code groups to classify assemblies so .NET can determine which security permissions to grant. A code group binds a single permission set to a single evidence (see Figure 1). An assembly must satisfy a code group's evidence to win the permissions in the permission set associated with the code group.

Code groups aggregate into .NET security policies. The permissions granted by a policy to a given assembly is the union of all permissions granted by the code groups in that policy whose evidence the assembly satisfies. For example, if an assembly satisfies the evidences of code groups A, B, and C, but not the evidences required by code groups D and E, it is granted only the union of permissions A, B, and C (see Figure 2).

Compose Security Policies

.NET lets you provide multiple security policies with different scopes. You can apply restrictive policies to specific cases, such as individual users or machines with limited privileges, and you can apply more permissive policies to other machines and users.

One policy may grant an assembly permission denied by another policy. All policies must concur on the allowed permissions, though, so the actual permissions .NET grants to an assembly make up the intersection of all permissions granted by all the security policies.

Security policies have four types (or levels): Enterprise, Machine, User, and Application Domain. An Enterprise policy affects all machines in the enterprise. Each machine has a Machine policy specific to that machine. The User policy affects one user, and the Application Domain policy applies only to code running in a specific application domain.

You configure Application Domain policies only programmatically. Typically, you place more restrictive policies downstream and more liberal policies upstream. This

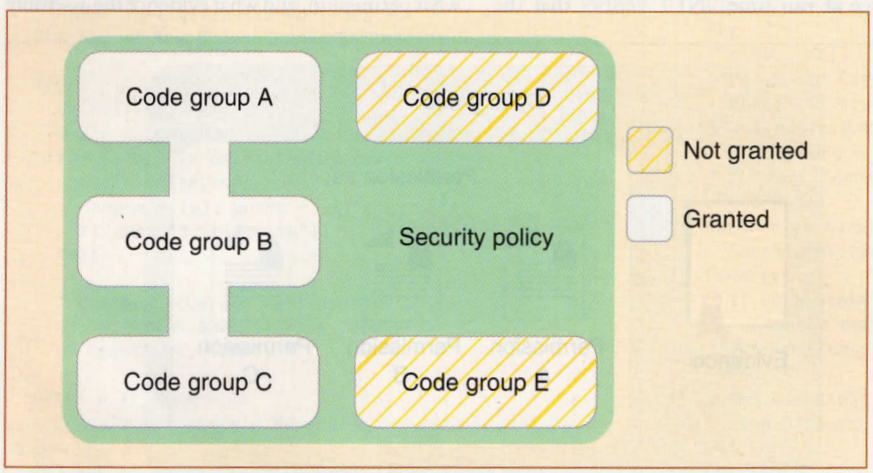

Figure 2 Implement Security With Policies. Code groups aggregate into .NET security policies. The permissions granted by a policy to a given assembly comprise the union of all the individual permissions granted by the code groups in that policy whose evidence the assembly satisfies. For example, if an assembly satisfies the evidence of code groups A, B, and C, but not the evidence required by code groups D and E, it is granted only the union of permissions A, B, and C.

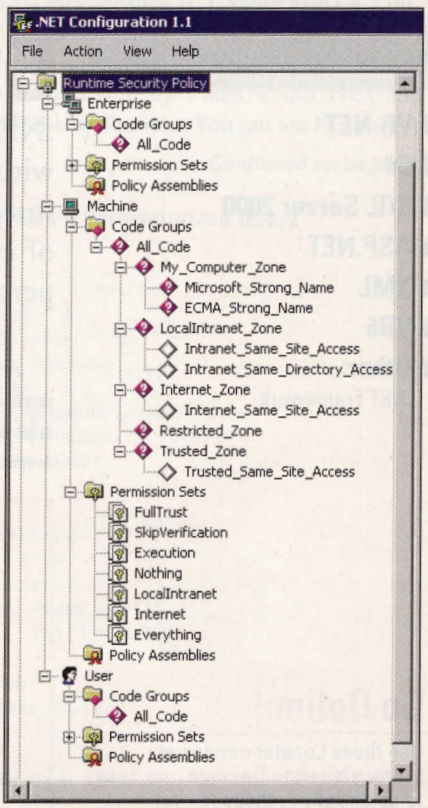

Figure 3 Administer Security With the .NET Configuration Tool. The common language runtime's code access security system determines an assembly's permissions to access protected resources. Each permission set granted to an assembly is based on the assembly's evidence (such as its URL or strong name). Each policy has a folder for permission sets and code groups, as you can see in the Runtime Security Policy folder.

allows overall flexibility with granular security policy—tight in some places and looser in others.

When .NET loads an assembly, it computes the permissions to grant that assembly. For each security policy, .NET aggregates the permissions from the code groups satisfied in that policy, then intersects the policies to find the combined overall collection of permissions to grant the assembly. It calculates that set of permissions only once (per app domain). The calculated permission persists in memory as long as the assembly remains loaded.

Whenever an assembly calls a .NET Framework class, that class may demand assurance from .NET that the assembly calling it has the security permission required to access it. For example, file I/O classes demand appropriate file I/O permission for that access. .NET throws a security exception if the assembly lacks the appropriate security permission.

So far, so good. But suppose a malicious assembly, which lacks permissions to access a class such as FileStream, calls a benign trusted assembly to do its dirty work. The .NET architecture can thwart such attacks. Whenever a class demands security permission checks, .NET traverses the entire call stack, making sure every assembly up the call chain has the required permissions. This is called the *security permission stack walk*. .NET aborts the stack walk immediately and throws an exception if it finds an assembly without permissions.

.NET gives you two ways to configure a code access security policy: caspol.exe, a command-line utility; or the .NET Configuration tool. Both have comparable capabilities. Use caspol.exe to make dynamic changes during installation, and use the .NET Configuration tool to configure security and export the security policies to other machines.

Administer Your Security Policies

The .NET admin tool has a Runtime Security Policy folder that contains an item for each policy: Enterprise, Machine, and User. Each policy item has subfolders containing its code groups and permission sets. Each security policy is stored in a dedicated XML-formatted security configuration file. The .NET Configuration tool merely provides a visual editor for these files.

The Permission Sets folders in all three policies contain the same set of named permission sets. Both the Enterprise and the User policies contain a single code group, All_Code, by default. This group uses the All Code evidence with the FullTrust permission set, so all assemblies are unrestricted.

The Machine policy has a single root code group, also called All_Code, that uses the All Code evidence with the Nothing permission set. It grants nothing by itself, and instead relies on the nested code groups to grant permissions.

The My_Computer_Zone code group uses the Zone evidence, with the zone set to My Computer, granting the FullTrust permission set. Consequently, all code coming from the local computer gets full trust. The My_Computer_Zone code group has two nested child code groups, Microsoft_Strong_Name and ECMA_Strong_Name (see Figure 3). These nested code groups use the Strong Name evidence (with the value set to the Microsoft public key) and ECMA public key, respectively. Both these nested code groups use the FullTrust permission set, granting unrestricted access to any assembly originating from Microsoft or ECMA, regardless of its zone.

The LocalIntranet_Zone code group uses the zone evidence with a value set to the Local Intranet zone. The permission set is LocalIntranet. The problem is that the LocalIntranet permission set doesn't grant Web access or file I/O permission. An assembly originating in the intranet might not function properly if it requires access to its original site or its install directory.

To compensate, the LocalIntranet_Zone code group contains two nested code groups. The Intranet_Same_Site_Access code group lets code access its site of origin, and the Intranet_Same_Directory_Access code group lets code access its original install directory.

The Internet_Zone code group uses the zone evidence, with a zone set to Internet. The permission set used is Internet. The Internet_Zone code group has a child code group called Internet_Same_Site_Access, allowing code coming from a site to connect to its site of origin.

The Restricted_Zone code group uses the zone evidence with a zone set to Untrusted Sites and the permission set to Nothing. The Trusted_Zone code group uses the zone evidence with a zone set to Trusted Sites. The permission set used is the Internet permission set. The Trusted_Zone code group has a child code group called Trusted_Same_Site_Access granting code coming from a trusted site permission to connect to its site of origin.

Build Custom Permission Sets

You can define custom permission sets and compose granular permissions suitable for a particular need. Create a new permission set by right-clicking on the Permission Sets folder and selecting New to bring up the Create Permission Set wizard. The first screen lets you name the new permission set and provide a description. Click on Next to assign individual permissions to the new permission set.

When you add a permission type from the left pane, a dedicated dialog for that type appears, allowing you to add individual permissions of that type. Suppose you need the new permission set to grant file I/O permissions to read the C drive, and full access to C:\temp. Select File I/O on the left pane of the dialog, and click on Add to bring up the file I/O permission setting dialog.

The dialog features a grid where each line corresponds to a single file I/O permission. You can also grant unrestricted access to the file system. Configure the required setting and click on OK to return to the previous dialog; click on Finish. You can now use this permission set with any code group in the policy.

You can modify existing code groups or create new ones. You might want to create a new code group that grants all assemblies signed with your organization's public key the Everything permission set. This can be handy when different teams use each other's assemblies across an intranet. The logical place for this new code group is in the Machine policy under the My_Computer_Zone code group. Highlight the My_Computer_Zone code group, click on it, and select New from the context menu. Name the new code group in the Create Code Group dialog, and click on Next.

Go to the condition type textbox in the "Choose a condition" type dialog. Select Strong Name from the drop-down combo box. This changes the lower part of the dialog to reflect the value of the requested strong key. Provide the public key value by importing it from an already signed assembly. Click on the Import button to browse to a signed assembly (either an EXE or a DLL) and select it. The wizard reads the public key from the assembly's manifest and populates the Public Key textbox.

Click on Next to proceed to the next dialog, where you need to assign a permission set to the new code group. You can use any existing permission set in the policy by selecting the set from the drop-down combo box. Select Everything, click on Next, and click on Finish in the next dialog. The new code group is now part of the policy. VSM

Juval Löwy is a software architect and principal of IDesign, a .NET consulting and training company. He's also a Microsoft regional director, the .NET California Bay Area User Group program committee chairman, and a conference speaker. This article derives from his upcoming book on .NET components (O'Reilly & Associates). Contact Juval at www.idesign.net.

Additional Resources

- Microsoft .NET Framework Security Overview: http://msdn.microsoft.com/vstudio/techinfo/articles/developerproductivity/frameworksec.asp
- .NET and Security: www.microsoft.com/net/technical/security.asp
- Code Access Security: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconcodeaccesssecurity.asp