Interested in learning more about security?

SANS Institute InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Code Access Security and Policy in Microsoft's .NET

.NET is Microsoft's new platform, oriented towards Internet-based applications and Web services. Because of its orientation towards programs loaded and run from the Internet, it was designed with security in mind. At runtime, the.NET framework can determine what permissions to allow to a block code depending on evidence, which includes the location of the code (local disk, intranet, internet, etc.) and its publisher. This allows the runtime system to give trusted code full access, and allowing code from an untrustworth...

Copyright SANS Institute

Author Retains Full Rights

AD

DEEPARMOR®

Code Access Security and Policy in Microsoft's .NET
Steven Kovner
January 1, 2003
GSEC Practical, Version 1.4b, option 1

1 Abstract

.NET is Microsoft's new platform, oriented towards Internet-based applications and Web services. Because of its orientation towards programs loaded and run from the Internet, it was designed with security in mind. At runtime, the.NET framework can determine what permissions to allow to a block of code depending on evidence, which includes the location of the code (local disk, intranet, internet, etc.) and its publisher. This allows the runtime system to give trusted code full access, and allowing code from an untrustworthy source to execute, while preventing it from performing operations which could cause damage.

The system administrator can assign code to different code groups, and control the permissions allowed to each group, by setting the .NET framework security policy.

This document gives an overview of .NET security; explains how evidence-based security works; and gives information and suggestions for setting the security policy.

2 Overview of .NET

.NET is Microsoft's solution for delivering Web services and components via the Web, as well as the new APIs and runtime systems for programs running locally on Windows systems. As described on Microsoft's web site:

> Microsoft® .NET is a set of Microsoft software technologies for connecting your world of information, people, systems, and devices. It enables an unprecedented level of software integration through the use of XML Web services: small, discrete, building-block applications that connect to each other—as well as to other, larger applications—via the Internet.[1]

.NET is composed of several components:

- The .NET Framework
- Developer Tools
- Servers
- Client Software

Refer to Microsoft's [.NET Framework] Product Overview for more information.

The .NET framework is the only component of interest in this paper.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

2.1 Framework

The .NET framework is the runtime system for .NET. It includes the Common Language Runtime (CLR) and the Class Libraries.[2]

2.1.1 Common Language Runtime (CLR)

The CLR is what actually loads and executes .NET code. Code for .NET is compiled to an intermediate language (IL) and compiled to native machine code by a just-in-time (JIT) compiler. Code in IL is called Managed Code. The runtime can determine what managed code will do, so that it can prevent managed code from performing operations it does not have permission for, unlike unmanaged native code.

The CLR validates each section of code before execution, and makes sure it does not perform operations it is not allowed to do. This is the core of .NET security.

Note that IL code can still call native (unmanaged) code, but this requires special permission, as it opens holes in the security model. Pure .NET applications will not need to call unmanaged code, but the CLR does call unmanaged code to make system calls. Code that needs access to legacy (pre-.NET) libraries can be written, but will require special permission to run.

2.1.2 Class Libraries

.NET programs need some way of accessing the Win32 API, as well as other functions supplied by the .NET runtime. Because .NET and its languages (VB.NET, C#, Managed C++) are object-oriented, .NET supplies a set of classes which allow access to system and .NET functions in an object-oriented manner. The class libraries that ship with .NET support both the Win32 functions as well as additional .NET functions, and they do this in a way that is compatible with the .NET security model. Microsoft describes the functions of the class libraries as follows:

> Base classes provide standard functionality such as input/output, string manipulation, security management, network communications, thread management, text management, user interface design features, and other functions.[3]

The complete list of classes is quite long, and includes classes that access system functionality, for example I/O, network I/O, and threads; as well as classes that make programming easier, such as string manipulation and collections.

Because the Win32 API is not secure, the class libraries implement the .NET security model. Managed code cannot call the API directly, without special permission. Normally, only code on the local filesystem has this permission. However, code from other locations can still access the API functions via the .NET class libraries, which can check for more specific permissions.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

2.2 Assemblies

Assemblies are groups of one or more files, comprising executable code (executable files or dynamic link libraries) and the metadata needed to describe them. They are the basic installation units of .NET. All code is shipped as part of an assembly. The metadata in an assembly includes security information used by the runtime system; this information is part of the evidence used by evidence-based security.

3 Introduction to .NET security

Microsoft designed .NET with security in mind. As Glen Kunene, reporter for DevX said, "And unlike past Microsoft platforms, .NET shows evidence of having been designed with security demands in mind."[4]

3.1 Types of security

The security architecture of .NET is made up of several components:[5]

- Evidence-based security
- Code access security
- The verification process
- Role-based security
- Cryptography
- Application domains

3.1.1 Evidence-Based Security

The CLR uses evidence to assign a piece of code to a code group. The evidence describes the origins of the piece of code. Seven types of evidence are defined, but administrators can add more. The seven types are Site, URL, Zone, Application Directory, Strong Name, Publisher, and Hash.[6]

The first four describe the location the code was loaded from; the next two describe who wrote the code; and the final evidence allows determining a particular build of the code.

To make assigning permissions easier and more logical, instead of assigning permissions directly based on the evidence, the CLR uses the evidence to determine the code group to which a piece of code belongs. Code groups are used to group together code that meets similar conditions, such as where they came from.[7] The code groups each have a single membership condition, and a single permission set.

Each code group can contain only one permission set; therefore permissions are grouped into permission sets.

© SANS Institute 2003, Author retains full rights.

The groups are arranged in a hierarchy. To determine which code group a piece of code belongs to, the CLR walks the hierarchy checking the membership conditions against the evidence of the piece of code. If the evidence matches the membership condition for a code group, that piece of code is given the permissions contained in the permission set assigned to the code group.[8] Code groups will be described in more detail below.

.NET ships with predefined code groups and permission sets, but the administrator can add more and modify or delete (most of) the existing ones. This will be covered more in the description of policy.

3.1.2 Code Access Security

The enforcement mechanism for evidence-based security is code access security.[9] As described above, when code assemblies are loaded, the CLR assigns them permissions via the evidence-based security mechanism.

When code in an assembly attempts to access a protected resource, the CLR will determine whether or not that assembly and all calling assemblies have the necessary permissions. This is done by walking the stack. The CLR checks each assembly in the call stack to determine whether or not it has the requested permission. If not, a security exception is thrown. If all assemblies have the necessary permission, permission is granted and execution continues. Note the stack walk can be a performance hit, but it is necessary to prevent a "luring" attack, in which malicious code calls trusted code and tricks it into performing an action for which the malicious code does not have permission.[10]

Code can "short-circuit" the stack walk by using the assert method of a permission object. This is needed so that the runtime can call the unmanaged code of the underlying operating system without giving the untrusted callers that permission. Only highly trusted code that has the assert permission can do this. Assertion must be used with care as it can open up security holes.[11]

Also note the permission check during the stack walk needs to be performed only when the assembly changes; calls within an assembly need not be checked because the permissions are granted on an assembly basis. Therefore, programmers can make this more efficient by designing their code to minimize calls between assemblies.[12]

Assemblies can specify in advance the permissions they need. This is called declarative permission checking. These checks are performed at JIT compile time, and allow an assembly to specify minimum, maximum, and refused permissions. The CLR will not load the assembly if it cannot grant it the minimum permissions. Refusing permissions allows code to restrict its operations and the operation of all code it calls, even if it would normally have permission to perform certain operations.

At runtime, code can also request or refuse permissions, allowing code to handle security exceptions in controlled locations.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

3.1.3 The Verification Process

Managed code is verified just before it is run, at JIT-Compile time. This verifies type safety, verifying an object is not accessed as an object of a different type; verifies that an integer is not accessed as a pointer; private fields are not accessed from outside the class; checks for buffer overflows; and other common programming bugs. These bugs often result in security holes. [13]Relying on a compiler to prevent them still allows malicious machine code to be executed if it can be inserted into the program. Performing these checks just before execution prevents any execution of unchecked code.

Note that individual classes and individual methods within a class are not loaded until they are needed, so the checks that are run at JIT-compile time may not occur until some later time during execution.[14]

For a more detailed verification of loading .NET applications and the security checks that are performed, see LaMaccia, part 2, pages 4-7, and part 3, pages 1-2.

3.1.4 Role-Based Security

This is the portion of the .NET security framework that determines who the user is on whose behalf the work is being done, and determines what he is authorized to access. That is, role-based security handles authentication and authorization.[15]

Role-based security will be covered at a high level; details are beyond the scope of this document.

3.1.4.1 Authentication

Authentication is the process of determining the user identity. The .NET Framework supports several methods of authentication:[16]

- Forms-based authentication. This uses HTML forms that the user fills out. ASP.NET then issues a cookie containing the credentials.
- Passport Authentication. This uses Microsoft's Passport to perform authentication.
- IIS. This uses any of the methods IIS uses to support authentication, including Basic Authentication, NTLM, Kerberos, Digest Authentication, and X509 Certificates (using SSL).
- Windows Authentication. This uses Window's authentication mechanism, including Kerberos, NTLM, and X509 Certificates.

Developers can also write custom authentication mechanisms.

3.1.4.2 Authorization

Authorization is the process of granting access to resources. .NET supports File Authorization and URL Authorization.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

File authorization is used with Windows Authentication, and verifies that the file's ACL allows the user access to the file.

URL authorization allows .NET programs to perform role checks at runtime to allow access to URLs, with additional control over the request method (GET, HEAD, POST, etc.)

3.1.4.3 Principal and Identity

"A Principal represents the security context under which the code is running while an Identity represents the identity of the user associated with that security context."[17]

The identity is generally created via the authentication. It is then attached to a Principal which is then associated with an execution context. The code can then obtain the identity roles from the principal and allow permissions according to the roles.

3.1.5 Cryptography

.NET also supports cryptographic primitives for encryption, digital signatures, hashing, and random number generation.[18] .NET supports the most popular algorithms for public key and private key cryptography, and for hashing. Supported encryption algorithms include RSA and DSA for public key encryption; DES, 3DES, and RC2 for private key encryption, and MD5 and SHA1 hashing algorithms.

3.1.6 Application Domains

Application domains allow an additional security boundary, allowing multiple applications to run within a process, with a security boundary between all the application domains. This is made possible with managed code, because the CLR prevents such operations as using integers as pointers or pointer arithmetic, which would allow access to one application domain to another.

4 Security policy overview

Without policy, the .NET security mechanisms described are not very useful. .NET gives the system administrator great flexibility in setting up an appropriate policy, allowing him to define and use code groups to allow permissions exactly as needed.

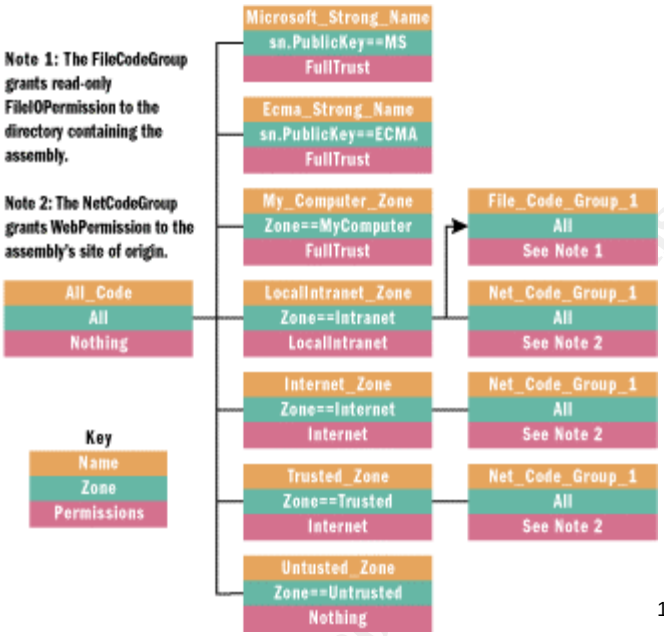
Since the policy is based on code groups and permission sets, they will be described first. This will be followed by a description of policy levels and management.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

4.1 Code Groups

As described in section 3.1.1, a code group is defined by a membership condition; that is the evidence needed to determine whether a piece of code belongs to the group or not; and a permission set. These are arranged hierarchically. The runtime will search the hierarchy, finding all code groups that a particular piece of code belongs to, and then assign that code the union of all permission sets of the code groups to which it belongs.

.NET ships with several code groups:



19

This diagram shows, for each group, the name; the evidence; and the permission set for each code group. The hierarchy starts with the All_Code group, which has no permissions. Below this are groups based on different types of evidence. The Microsoft_Strong_Name and Ecma_Strong_Name groups match code that is digitally signed by Microsoft and ECMA, respectively. These are granted full trust. The other groups at this level are based on the security zones; these are the same zones assigned by Internet Explorer. Note that code on the local machine has full trust; code from other locations has less permission. NOTE – as of .NET Framework Service Pack 1, code in the Internet_Zone does not have Internet permissions; it has the Nothing permission set by default.[20]

The File_Code_group_x and Net_Code_group_x permission sets are dynamically generated. The file code group allows code read-only access to the directory from which it was loaded; and the net code group allows code access to the URL from which it was loaded.[21]

System administrators can create their own code groups. Generally this would be done to allow greater access to code from particular publishers, or to specific programs. For example, additional permissions could be given to web

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

services code from a particular publisher. Normally these groups would be subsets of the All_Code group, although subsets of other groups are allowed, for example, if creating a subset of the Trusted_Zone to grant greater permissions to code from a particular URL. (Note that since web sites can change, it is better to grant permissions based on cryptographic evidence, such as publisher.)

Code groups can be marked as "exclusive". This prevents the runtime from adding any other permissions that also match the evidence presented by a piece of code. This is used to exactly specify the permissions granted a piece of code while preventing permissions being granted from any other code group at any level.[22] Note that the runtime will not allow an assembly to execute if it belongs to more than one exclusive code group, so any exclusive code groups must be set up to match a fairly restrictive set of evidence.[23]

4.2 Permissions and Permission Sets

There are many individual permissions within the .NET framework. Some of these are: (all descriptions are quotes)[24]

- EnvironmentPermission – the ability to read and write environment variables
- FileDialogPermission – the ability to access files that have been selected by the user in the Open dialog box
- FileIOPermission – the ability to work with files…
- SecurityPermission – the ability to execute, assert permissions, call into unmanaged code, skip verification, ad other rights.
- UIPermission – the ability to access the user interface.

The complete list is available in Robinson and in the .NET framework documentation.

Note that some of these are fine-grained. It is possible to allow FileIOPermissions for particular files or particular directories, and control read and write permission separately. UIPermission can allow or deny access to the Clipboard separately from other permissions.

Since code groups each can allow only one permission set, individual permissions are grouped into permission sets which are then granted as a whole via the code group mechanism.

The .NET framework ships with several predefined permission sets:

Permission Set	Description
Nothing	The empty permission set (grants nothing)
FullTrust	Implicitly grants unrestricted permissions for all

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Permission Set	Description
	permission types
Everything	Explicitly grants unrestricted permissions for all built-in permission types
SkipVerification	SecurityPermission: SkipVerification
Execution	SecurityPermission: Execution
Internet	FileDialogPermission: Open IsolatedStoragePermission: DomainIsolationByUser (quota = 10240) UIPermission: OwnClipboard | SafeTopLevelWindows Printing Permission: SafePrinting
LocalIntranet	SecurityPermission: Execution | Assert | RemotingConfiguration FileDialogPermission: Unrestricted IsolatedStoragePermission: AssemblyIsolationByUser (no quota) UIPermission: Unrestricted PrintingPermission: DefaultPrinting EnvironmentPermission: Read-only (USERNAME/TMP/TEMP) ReflectionPermission: ReflectionEmit DNSPermission: Yes EventLog: Instrument

Built-in Permission Sets [25]

As can be seen from this, the permissions can be fine-grained to allow programs just the permissions they need to execute, yet restrict their abilities to modify other parts of the system.

Note that the Internet permission set is not applied to code from the Internet zone, as of the .NET framework SP1. Code from the Internet zone has the “Nothing” permission set.

Permission sets may be created and modified by the system administrator. However, of the built-in sets, only the Everything set may be modified. This allows, for example, a particular application to have access only to the data that it needs, and not to any other files on the computer.

4.3 Policy levels

The .NET security policy is set at four levels. From highest to lowest, they are:[26]

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

- Enterprise
- Machine
- User
- AppDomain

The Enterprise level sets the policy for an entire enterprise. While it is possible to have different enterprise policies on different machines, it is not advisable. This level is set by enterprise administrators. By default, this is set to unrestricted.

The machine policy is set by system administrators for a particular machine. By default, this contains all the actual security settings; the others are all set to unrestricted.

The user policy is set by individual users. By default, this is set to unrestricted.

The AppDomain policy controls settings for an application domain host. An "application domain host" is software "that creates the application domain and loads assemblies into it."[27] These hosts are executables that load the managed code and runtime. They include the shell host that loads .exe files; browser hosts, such as Internet Explorer, and server hosts, such as the ASP.NET server. This level can only be set programmatically[28], so it is not normally administered and will not be discussed further.

The permissions granted to a piece of code is the intersection of the permissions granted by all these levels. Therefore, if any one level denies permissions, the code does not get that permission, and a user may not grant permissions denied at the machine or enterprise level by an administrator. A user can, however, further restrict permissions and break applications, so it is advisable to prevent users from changing the .NET security policy.

A level can be marked "final", which will stop the runtime from evaluating lower-level security policies. Thus, an administrator can mark the Enterprise level as final, which will prevent any other level from being checked.[29]

4.4 Policy files

The Enterprise, Machine, and User policies are contained in XML files. The Enterprise and Machine policy files are in
%SYSTEMROOT%\Microsoft.NET\Framework\<version>\CONFIG
where <version> is the current .NET Framework version. The version v1.0.3705 is current as of this writing. Note that several versions of the framework can coexist, and that they all need to be configured if policy changes. The enterprise settings are in enterprise.config and the machine settings are in security.config.

The user settings are in %USERPROFILE%\application data\Microsoft\CLR security config\<version>\security.config .

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

While these files can be directly edited, Microsoft has supplied software to make setting policy much easier.

4.5 Policy Management

Microsoft supplies two tools for managing .NET security: an MMC (Microsoft Management Console) snap-in, and caspol.exe.

The mscorcfg.msc snap-in for MMC supplies a console for managing .NET configuration as well as some security wizards. It is easy to use, but controls only a single system. However, it can be used to create a Microsoft Installer package which can be deployed to other systems (see below).

The caspol.exe program is a command-line program. It is harder to use, but scripts that use it can be created and distributed.

For detailed information on managing security policy with Microsoft tools, refer to http://msdn.microsoft.com/library/en-us/cpguide/html/cpconsecuritypolicyconfiguration.asp .

One problem with caspol.exe is that it can be run by non-administrators. A caspol.exe command can be used to turn off security checking entirely. Obviously, this is a large security hole. Administrators should set permissions on the caspol.exe file to allow only administrators to execute the file.[30]

4.5.1 Increasing Permissions for a Specific Application

If there is a trusted application that does not run because required permissions have not been granted, create a custom code group and permission set for it. This can be done using the following steps:[31]

1. The permview.exe tool will display the permissions the assembly requests. Run this tool on the .exe and .dll's that make up the application.
2. Determine something that can uniquely identify the assembly, such as its strong name, hash, public key, or a custom attribute. Make sure this is unique, as all code that has this characteristic will be trusted.
3. Create a custom code group with this characteristic as its membership condition, using mscorcfg.msc or caspol.exe.
4. Create a custom permission set including the necessary permissions.
5. Assign the new permission set to the new code group.
6. Deploy the new code group and permission set.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Note that the permview.exe tool displays the minimum, optional, and refused permissions, if the assemblies specified them declaratively. Applications may not specify the permissions they need in this manner, or the declarative permissions may not include all the permissions needed. The latter case is a bug. If an application has these problems, the vendor should be able to help specify the actual permissions needed (and also fix the bug by requesting all the required permissions declaratively).

4.5.2 Deploying policy

Microsoft has made it relatively easy for an administrator to set and deploy a policy across an enterprise.

The MMC snap-in allows a system administrator to create a policy deployment package in a Microsoft Installer file and distribute it across an enterprise via the group policy editor or Systems Management Server. Use of group policy requires all systems to be running Win2000 or later. For more information, refer to ".NET Framework Enterprise Security Policy Administration and Deployment" at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/entsecpoladmin.asp and GotDotNet's article on "Security Policy Best Practices" at http://www.gotdotnet.com/team/clr/SecurityPolicyBestPractices.htm .

Alternatively, the administrator can write a script that uses caspol.exe and distribute that.

If Microsoft Installer files are used, the changes can be rolled back by uninstalling the change. But when installing or uninstalling, the administrator must make sure the account has permissions to modify the configuration files. The Windows Installer will not produce an error if the user does not have permission to change the configuration file.[32]

5 Tests and Results

This writer ran several tests to determine the permissions granted to a .NET program under different circumstances. I wrote a simple program in C#, which puts up a dialog box, with two labels indicating whether the program has FileIO permission for drive C:, and whether it has all possible SecurityPermissions. The code for this program is in Appendix A. There also is a web page which attempts to download and run the program when a link is clicked. The web page is in Appendix B.

The test environment comprised two systems: a system running Windows XP Pro; and a system running Windows 2000 Pro. Both had Internet Explorer 6, Visual Studio .NET, and the latest version of the .NET runtime installed. They were also both up-to-date with Microsoft's patches.

I tested the program by running it on the local drive; on a shared drive; by emailing it; and by downloading it from a web site. In the latter two cases, I tested the program run from its current location, and also by downloading it first. Note

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

that for web testing, I was only able to perform the test on the local intranet, as I do not have access to a web server on another network.

When run from the local drive, the program ran and had both FileIO and Security permissions.

I shared the drive it was on and mounted it on the Win2000 system. It ran, but did not have either FileIO or Security permission, as it was running on the local intranet.

I emailed the program to myself. One system was set up to disallow running any scripts or executables, so the program could not be run. On the other, when I attempted to run it from the email, I had to proceed past more than one warning dialog box, but the program still would not run; it failed with an exception indicating it lacked Execute permission. Saving the file first, and running it from the location it was saved, allowed it to run with both FileIO and Security permissions, because it was now run from the local drive. This means that a user can still be tricked into saving and then running a malicious application. It might even be possible for scripts to run that save the program and run it without the user's knowledge.

I then wrote a simple web page linking to the program, reproduced in Appendix B.

This was served from the Windows XP system. Due to my system configuration, I could only test this on systems on the local intranet. In those cases, the program ran, but without FileIO and Security permissions. I ran an additional test, putting the web site in the "restricted sites list" in Internet Explorer. In that case, the program could not be run. If .NET operates as documented, the program would also not run in the Internet zone.

Finally, I found that it is possible to limit the permissions granted to the test program even when run from a local disk, without breaking some .NET applications (Visual Studio .NET). I used mscorcfg.mmc to change the permission set for the My_Computer_Zone code group from "FullTrust" to "LocalIntranet". The program could still run, but was denied FileIO and Security permissions. Visual Studio .NET and the .NET runtime code are signed by Microsoft and/or ECMA, and therefore are members of the Microsoft_Strong_Name and the ECMA_Strong_Name code groups. I also set the permission set to "Nothing". Visual Studio .NET still ran, but the program I wrote could not execute at all.

Therefore, system administrators can still restrict users' abilities to download and run code by changing the permissions granted to the My_Computer_Zone, as long as only applications that are signed are used. Additional code groups can be added under the My_Computer_Zone, at the same level as the Microsoft_Strong_Name group, that grant specific applications the additional permissions they need. This will not always be possible; for example, developers will need to be able to run their own code before it is

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

signed, and a company may require use of specific applications that are not signed. But in many cases, this is a quite workable solution.

These results can be summarized in a table:

Location	Executes	Has FileIO permission	Has Security permission
Local drive with default security settings	Yes	Yes	Yes
Local drive with My_Computer_Zone granted only LocalIntranet permission set.	Yes	No	No
Local drive with My_Computer_Zone granted the "Nothing" permission set	No	No	No
Shared drive	Yes	No	No
Email – executed directly from email message	No	No	No
Email – saved from email message, and then run	Yes	Yes	Yes
Web site, local intranet	Yes	No	No
Web site, Intranet (untested; based on documentation)	No	No	No
Web site, Restricted Sites zone	No	No	No

© SANS Institute 2003, Author retains full rights

6 Recommendations

Other than changing the file protection on caspol.exe, .NET ships with a reasonable security configuration, but not a perfect one. It is a good starting point for further customization.

Security administrators need to consider an organization's security policy, its users, and its applications to define an appropriate .NET security policy. Some

settings will be common across most machines and some will be for specific machines (e.g. servers visible to the outside world, such as web servers).

An administrator may need to further restrict permissions depending on his organization's policy. If the policy requires users to only run approved software, and does not allow downloading, the My_Computer_Zone should have its permission set restricted, with specific applications granted the necessary permissions based on their strong names (signatures). The permission set granted to My_Computer_Zone can be set to "Nothing", which will allow only signed code to execute.

The policy at the Enterprise level must allow all the permissions needed anywhere in the company, because lower levels can only restrict permissions, they cannot grant more. Policies for specific users and for specific machines therefore need to restrict permissions for users and machines which do not need all the permissions granted at the enterprise level.

Specific applications or code from trusted vendors that is loaded from a network location may need a custom code group to grant them the needed permissions to execute.

7 Summary

Microsoft has done a good job in implementing security in the .NET framework, but not a perfect job. And, as with all security mechanisms, the degree of security obtained depends on the system administrators, programmers, and users. System administrators can leave their systems insecure; programmers can still write insecure code; and users can still bypass security.

While the default policy is adequate for a standalone system, system administrators must make sure they modify the policy as needed for their enterprise, and they must prevent local users from overriding the security settings by preventing them caspol.exe. The .NET framework allows system administrators to allow execution of only specific programs.

The .NET framework implements security mechanisms, but it is still up to the software developer to use them properly, and to not bypass them (e.g. with the Assert method).

Finally, users can still ignore all the warnings and download code which will then execute with full permissions, with the default security settings. System administrators can prevent this, if needed.

Appendix A – Test program

This is the C# source code for the security test program. Most of this is code generated by Visual Studio .NET; only a few lines in the SecurityTest constructor were written by hand.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Security;
using System.Security.Permissions;

namespace SecurityTest
{
    /// <summary>
    /// SecurityForm displays permissions available to itself.
    /// </summary>
    public class SecurityForm : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label LabelHasFileIO;
        private System.Windows.Forms.Label LabelHasSecurity;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public SecurityForm()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // Change the text on the forms if permissions not granted
            //
            // start with FileIO
            try
            {
                // this could be better by getting a list of drives and
                // checking access to all drives, but
                // for our purposes, checking drive C is sufficient.
                FileIOPermission fileIOperm = new
                    FileIOPermission(FileIOPermissionAccess.AllAccess,
                    @"C:\");
                fileIOperm.Demand();
            }
            catch
```

© SANS Institute 2003, Author retains full rights

```
    {
        this.LabelHasFileIO.Text = "FileIO permission denied";
    };

    // now test for security
    try
    {
        // Check for all security permission flags
        SecurityPermission secperm = new
            SecurityPermission(SecurityPermissionFlag.AllFlags);
        secperm.Demand();
    }
    catch
    {
        this.LabelHasSecurity.Text = "Security permission denied";
    };
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.LabelHasFileIO = new System.Windows.Forms.Label();
    this.LabelHasSecurity = new System.Windows.Forms.Label();
    this.SuspendLayout();
    //
    // LabelHasFileIO
    //
```

© SANS Institute 2003, Author retains full rights

```
            this.LabelHasFileIO.Location = new System.Drawing.Point(48, 24);
            this.LabelHasFileIO.Name = "LabelHasFileIO";
            this.LabelHasFileIO.Size = new System.Drawing.Size(136, 24);
            this.LabelHasFileIO.TabIndex = 0;
            this.LabelHasFileIO.Text = "FileIO permission granted";
            //
            // LabelHasSecurity
            //
            this.LabelHasSecurity.Location = new System.Drawing.Point(40, 64);
            this.LabelHasSecurity.Name = "LabelHasSecurity";
            this.LabelHasSecurity.Size = new System.Drawing.Size(144, 32);
            this.LabelHasSecurity.TabIndex = 1;
            this.LabelHasSecurity.Text = "Security permission granted";
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(224, 125);
            this.Controls.AddRange(new System.Windows.Forms.Control[] {

              this.LabelHasSecurity,

              this.LabelHasFileIO});
            this.Name = "Form1";
            this.Text = "Security Test";
            this.ResumeLayout(false);

        }
        #endregion

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.Run(new SecurityForm());
        }
    }
}
```

Appendix B – Test web page

This is the web page that allowed download of the security test program.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <TITLE></TITLE>
        <META NAME="GENERATOR" Content="Microsoft Visual Studio 7.0">
    </HEAD>
    <BODY>
        Test your .NET security settings.<br>
        <a HREF="SecurityTest.exe">Click here to test</a></BODY>
</HTML>
```

References

"About .NET security". GotDotNet. URL: http://www.gotdotnet.com/team/clr/about_security.aspx (December 17, 2002).

"Application Domain Host" Microsoft Corporation. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconapplicationdomainhosts.asp (December 12, 2002).

"Authentication and Security Mechanisms in ASP.NET Web Applications". Internet Security Systems. URL: http://documents.iss.net/whitepapers/asp_net_whitepaper.pdf (December 10, 2002).

Box, Don. "Security in .NET: The Security Infrastructure of the CLR Provides Evidence, Policy, Permissions, and Enforcement Services". MSDN Magazine, September, 2002. URL: http://msdn.microsoft.com/msdnmag/issues/02/09/SecurityinNET/default.aspx (December 18, 2002).

"Configuring Security Policy ".Microsoft Corporation. URL: http://msdn.microsoft.com/library/en-us/cpguide/html/cpconsecuritypolicyconfiguration.asp (December 12, 2002).

"Defining the basic elements of .NET". Microsoft Corporation. URL: http://www.microsoft.com/net/basics/whatis.asp (December 2, 2002).

DeJesus, Edmund X. "Secure Your Web Services Applications". .NET Magazine, June 2002. URL: http://www.fawcette.com/dotnetmag/2002_06/magazine/features/edejesus/ (December 20, 2002).

Foundstone, Inc. and CORE Security Technologies. "Security in the Microsoft _NET Framework". URL:

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/net/evaluate/fsnetsec.asp (December 10, 2002).

"How Do I … Script Security Policy Changes". VBCentral. URL: http://www.vbcentral.net/quickstart/howto/doc/security/SecScripting.aspx (December 10, 2002).

Kunene, Glen. "Software Engineers put .NET and Enterprise Java Security to the Test". DevX. URL: http://archive.devx.com/enterprise/articles/dotnetvsjava/GK0202-1.asp (December 20, 2002).

LaMacchia, Brian; Lange, Sebastian; Lyons, Matthew; Martin, Rudi; and Price, Kevin. .NET Framework Security Addison Wesley, April 2002. Excerpt read at URL: http://searchwebservices.techtarget.com/content/0,290959,sid26_gci856076,00.html (December 10, 2002).

Lippert, Eric. "Code Security in .NET". July 11, 2002. URL: http://www.ssdotnet.org/Custom/Presentations/20020711_Lippert/20020711_Lippert_Security.ppt (December 10, 2002).

McBee, Rob. "Microsoft .NET – An Overview". GIAC GSEC Practical, September 14, 2002. URL: http://rr.sans.org/win/MS_net.php (December 20, 2002).

"Microsoft .NET Framework Security Overview". Microsoft Corporation. URL: http://msdn.microsoft.com/vstudio/techinfo/articles/developerproductivity/frameworksec.asp (November 4, 2002)

"317339- INFO: .NET Framework Change in Default Machine Level Security Policy". Microsoft Corporation, November 30, 2002. URL: http://support.microsoft.com/default.aspx?scid=KB;en-us;317399& (December 20, 2002).

".NET Framework Enterprise Security Policy Administration and Deployment". Microsoft Corporation. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/entsecpoladmin.asp (December 10, 2002).

".NET Security Policy". IT World, March 19, 2001. URL: http://www.itworld.com/nl/nt2k_sec/03192001/ (December, 2002)

".NET Security Policy Model" IT World, March 26, 2001. URL: http://www.itworld.com/nl/nt2k_sec/03262001/ (December, 2002)

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

"Product Overview". Microsoft Corporation. URL: http://msdn.microsoft.com/netframework/productinfo/overview/default.asp (December 3, 2002).

Robinson, Simon; Cornes, Ollie; Glynn, Jay; Harvey, Burton; McQueen, Craig; Moemeka, Jerod; Nagel, Christian; Skinner, Morgan, and Watson, Karli Professional C# Wrox Press, Ltd., 2001. [Edition based on .NET beta, not the later edition].

"Security and the Microsoft _NET Framework". Microsoft Corporation. URL: http://msdn.microsoft.com/netframework/techinfo/articles/security/foundstone.asp (November 4, 2002)

"Security Policy Best Practices". GotDotNet. URL: http://www.gotdotnet.com/team/clr/SecurityPolicyBestPractices.htm (December 20, 2002).

Serwin, Sebastian. "The Microsoft's .NET strategy" WindowSecurity.com, July 19, 2002. URL http://www.windowsecurity.com/articles/The_Microsofts_NET_strategy.html (November 15, 2002).

"Sun and Microsoft Defend Their Respective Enterprise Platform Security". DevX. URL: http://www.devx.com/enterprise/article/9658 (December 20, 2002).

"Understanding .NET Security". C# Security. URL: http://www.c-sharpcorner.com/Tutorials/DotNetSecurityVP001.asp (December 10, 2002).

Footnotes

[1] "Defining the Basic Elements of .NET", p. 1.
[2] Product Overview.
[3] Product Overview.
[4] Kunene, P. 2
[5] Foundstone Inc., and CORE Security Technologies, p.1
[6] Box, p. 53.
[7] Robinson, p. 1068.
[8] Box, p. 58
[9] Foundstone Inc., and CORE Security Technologies, p.6
[10] Lippert, pp. 19-20
[11] Foundstone Inc., and CORE Security Technologies, p.6
[12] LaMaccia, Part 3, p. 3
[13] Foundstone Inc., and CORE Security Technologies, pp. 6-7
[14] LaMaccia, part 3, p. 2
[15] Foundstone Inc., and CORE Security Technologies, p. 7
[16] Foundstone Inc., and CORE Security Technologies, pp. 7-8
[17] Foundstone Inc., and CORE Security Technologies, p. 9
[18] Foundstone Inc., and CORE Security Technologies, p. 9

© SANS Institute 2003, Author retains full rights.

[19] Box, p. 59
[20] .NET Framework Change in Default Machine Level Security Policy.
[21] Box, p. 59.
[22] Security Policy Best Practices, p. 8.
[23] Security Policy Best Practices, p. 8.
[24] Robinson, p. 1075.
[25] Box, p. 60
[26] Security Policy Best Practices, p.2.
[27] Application Domain Hosts.
[28] Box, p. 56.
[29] Security Policy Best Practices, p. 8.
[30] Robinson, p. 1099.
[31] Security Policy Best Practices, p. 7.
[32] Security Policy Best Practices, p. 11.

© SANS Institute 2003, Author retains full rights

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: August 17th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Adelaide 2017	OnlineAU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced