

.Net Security – A bird's eye view

By Jonathan Levin
JL@HisOwn.com

(C) 2003, JL@Hisown.com - All Rights Reserved

Part I

Framework Security Features

(C) 2003, JL@Hisown.com - All Rights Reserved

.Net Features

- Tight Language Interoperability
- MetaData
- JIT Compilation
- Garbage Collection
- OOP
- Code Access Security
- Base Class Library
- Native Code Interoperability

(C) 2003, JL@Hisown.com - All Rights Reserved

.Net Security Mechanisms

- Role-based security
- Evidence-based, Code-Access security
- Isolated Storage Containers

- Cryptography

- Compiler Enhancements

(C) 2003, JL@Hisown.com - All Rights Reserved

RBS

Role Based Security

- The user is restricted by his or her profile/permissions
(In the Win32 model, granted by groups)
- Useful in situations where server access is required
(protect the server against its users)
- Problematic on workstations and home PCs
(where most users are local administrators)
- Problematic when it comes to mobile code
(again - most users are local administrators, and will run anything!)

(C) 2003, JL @Hisown.com - All Rights Reserved

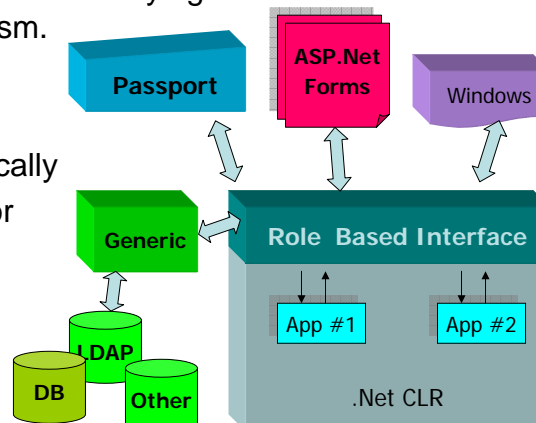
RBS

Role Based Security

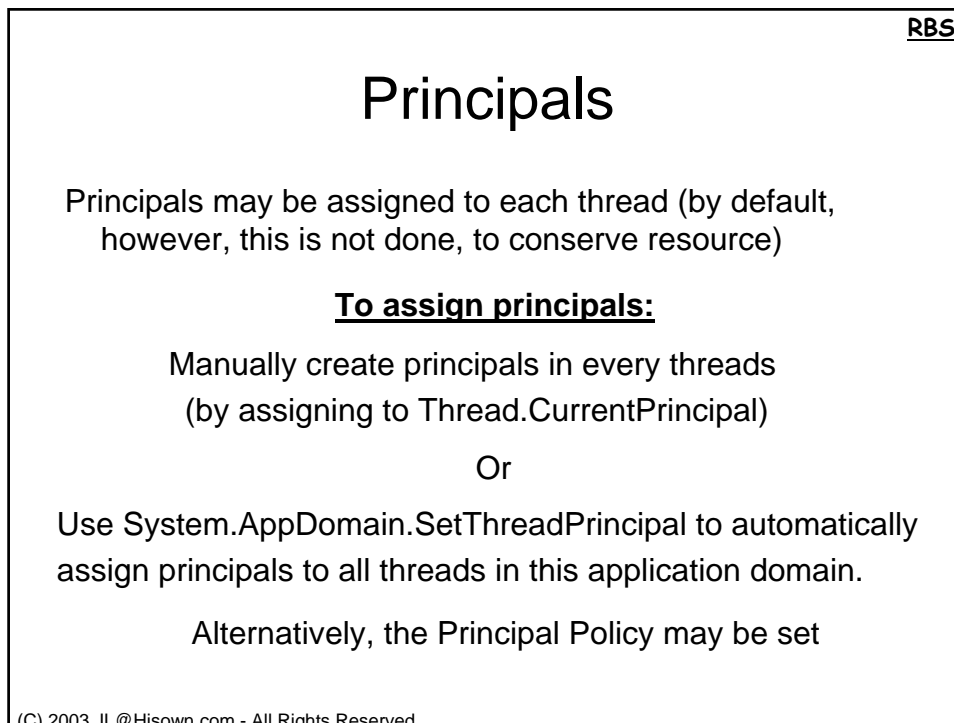
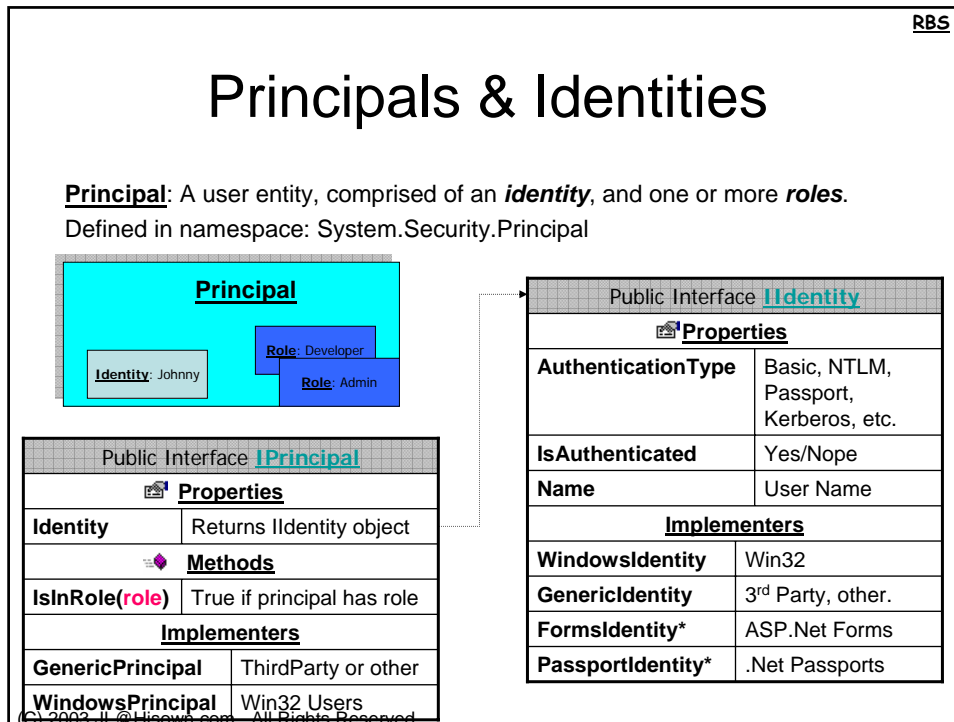
.Net defines a common Role Based interface

This interface defines four role “providers”, enabling the efficient abstraction of the underlying authentication and authorization mechanism.

The Generic provider is extensible, and is specifically designed for future and/or 3rd party providers.





(C) 2003, JL @Hisown.com - All Rights Reserved



RBS

Identities

The Windows Identity

System.Security.Principal: WindowsIdentity	
 Properties	
IsAnonymous	True if anonymous logon
IsGuest	True if guest logon
IsSystem	True if a system account
Token	Win32 Token
 Methods	
GetAnonymous()	Static. Returns an Anonymous WindowsIdentity object
GetCurrent()	Static. Returns the current WindowsIdentity object
Impersonate()	Enables thread to impersonate this identity


(C) 2003, JL @Hisown.com - All Rights Reserved

RBS

Identities

The ASP.Net Forms Identity

System.Web.Security: FormsIdentity	
 Properties	
Ticket	FormsAuthenticationTicket

System.Web.Security: FormsAuthenticationTicket	
 Properties	
CookiePath	Cookie Path= setting
Expiration	Date of Expiration
Expired	Boolean if Now() > Expiration
IsPersistent	False for session cookies
IssueDate	Date of first issue
Name	Cookie Name
UserData	Application Defined
Version	FFU

(C) 2003, JL @Hisown.com - All Rights Reserved

CAS

Code Access Security

Apply permissions on CODE executing,
which is identified by origin and creator.

Policies are accessible through: `System.Security.Policy.*`

Permissions tweaked using `System.Security.Permissions`

(C) 2003, JL@Hisown.com - All Rights Reserved

CAS

Permissions

<u>Permission</u>	<u>Applies to..</u>
EnvironmentPermission	Environment Variables
FileDialogPermission	Access to files/folders through a dialog
FileIOPermission	I/O to files and folders (System.IO.*)
IsolatedStoragePermission	Access to VFS
ReflectionPermission	(System.Reflection.*)
RegistryPermission	Microsoft.Win32.Registry
UIPermission	UI & Clipboard

(C) 2003, JL@Hisown.com - All Rights Reserved

Named Permission Sets

DnsPermission.Unrestricted
 EnvironmentPermission.Read(Username)
 EventLogPermission.Instrument
 FileDialogPermission.Unrestricted
 IsolatedStoragePermission.AssemblyIsolationByUser
 PrintingPermission.DefaultPrinting
 ReflectionPermission.ReflectionEmit
 SecurityPermission.Execution
 SecurityPermission.Assertion
 UIPermission.Unrestricted

FileDialogPermission.Open
 IsolatedStoragePermission.DomainIsolationByUser
 PrintingPermission.SafePrinting
 SecurityPermission.Execution
 UIPermission.SafeTopLevelWindows
 UIPermission.OwnClipboard

(C) 2003, JL@Hisown.com - All Rights Reserved

CAS

Specifying Security

Declarative Security:

During Compile time, specified in the assembly metadata

Imperative Security:

Enforced during run-time, by the CLR.

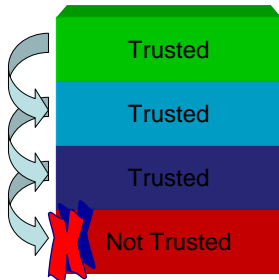
(C) 2003, JL@Hisown.com - All Rights Reserved

CAS

Do the Walk

To enforce permissions, the runtime “walks the stack”

If an untrusted assembly is encountered in the stack walk, a security exception is thrown, and permission is denied.



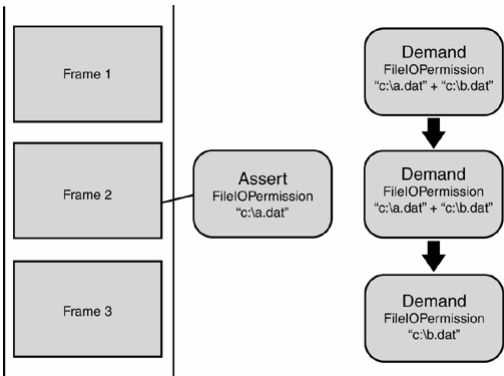
(C) 2003, JL @Hisown.com - All Rights Reserved

CAS

Assertions and Stack Walks

Using assertions, this behavior can be modified.

- Assert – enables
- Deny – denies
- Demand - walks



```
new FileIOPermission(FileIOPermissionAccess.Write, "requested_filename").Assert();
```

(C) 2003, JL @Hisown.com - All Rights Reserved

CAS

Demanding RBS permissions

Imperatively:

```
// In VB.Net – Use “Dim .. as PrincipalPermission”..
PrincipalPermission p1,p2;

p1 = new PrincipalPermission(“Johnny”, null);
p1.demand();
p2 = new PrincipalPermission(null, “Administrators”);
p2.demand();
```

Declaratively:

```
[PrincipalPermission(SecurityAction.Demand, Name = “Johnny”)]
[PrincipalPermission(SecurityAction.Demand, Role = “Administrators”)]
```

(C) 2003, JL @Hisown.com - All Rights Reserved

CAS

Forensics (obtaining evidence)

```
public void dumpEvidence()
{
    Assembly me;
    Evidence myEvidence;
    IEnumerator enumerator;
    /* Get a handle on moi, first...*/
    me = Assembly.GetExecutingAssembly();
    myEvidence = me.Evidence;
    enumerator = myEvidence.GetHostEnumerator();
    Console.WriteLine("Host Evidence:");
    /* Enumerate evidences supplied by host... */
    while (enumerator.MoveNext()) {
        Console.WriteLine("Evidence:\n" + enumerator.Current + "\n");
    }
    Console.WriteLine(Environment.NewLine);
    Console.WriteLine("Assembly Evidence:");
    enumerator = myEvidence.GetAssemblyEnumerator();
    while (enumerator.MoveNext())
    {
        Console.WriteLine(enumerator.Current + "\n");
    }
} /* end dumpEvidence */
```

(C) 2003, JL @Hisown.com - All Rights Reserved

CAS

Evidence Based Security

- **From what site was the assembly obtained?**
Assemblies are the building blocks of .NET Framework applications. They form the fundamental unit of deployment, version control, reuse, activation scoping, and security authorization. An application's assemblies are downloaded to the client from a Web site.
- **From what URL was the assembly obtained?**
The security policy requires the specific address from which the assembly was downloaded.
- **From what zone was the assembly obtained?**
Zones are descriptions of security criteria, such as Internet, intranet, local machine, and so forth, based on the location of the code.
- **What's the strong name of the assembly?**
The *strong name* is a cryptographically strong identifier provided by the author of the assembly. While it does not provide any authentication of the author, it uniquely identifies the assembly and ensures that it has not been tampered with.

(C) 2003, JL @Hisown.com - All Rights Reserved

CAS

Policy Evaluation

The screenshot shows the .NET Framework Configuration console. The left pane displays a tree view of the configuration hierarchy, including My Computer, Assembly Cache, Configured Assemblies, Remoting Services, Runtime Security Policy, Enterprise, Machine, and User. The right pane displays the 'Code Access Security Policy' page, which contains the following text:

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 publisher certificate), which in turn is based on configurable security policy.

To learn more about the code access security model, refer to the Microsoft .NET Framework SDK documentation.

The wizards and task links below will help you set and distribute security policy. For complete control of security policy, use the tree views under this node.

Tasks

- [Increase Assembly Trust](#)
Use the Trust an Assembly wizard to increase the level of trust granted to a particular assembly. This wizard modifies security policy based on information about the evidence of the assembly selected.
- [Adjust Zone Security](#)
Use the Security Adjustment Wizard to modify the level of trust granted to all assemblies coming from a particular zone, such as Internet, Local Intranet, or My Computer.
- [Evaluate Assembly](#)
Use the Evaluate an Assembly wizard to evaluate what permissions or code groups apply to a particular assembly. This is useful in determining the effect of current security policy on actual assemblies.
- [Create Deployment Package](#)
Use the Deployment Package Wizard to create a policy deployment package. The wizard wraps a security policy level into a Windows Installer Package (.msi file) that can then be distributed using Group Policy or Microsoft Systems Management Server.
- [Reset All Policy Levels](#)
Use this task to reset your security policy to the default settings. Note that this will obliterate all modifications that might have been made to default security policy.

(C) 2003, JL @His

IS

Isolated Storage

- A Virtual file system, unique to each assembly.
- A set of types & methods supported by the Framework for local storage.
- Each assembly is given access to a segregated storage on disk.
- No access to other data is allowed. Isolated storage is 100% private
- No need for file system path determination
- Access to isolated storage is restricted by zone:
 - Internet Zone: small quota
 - Intranet Zone: larger quota
 - Restricted Sites: No access

(C) 2003, JL@Hisown.com - All Rights Reserved

IS

Isolated Storage

Isolated storage may be SCOPED:

System.IO.IsolatedStorage.IsolatedStorageScope:

A bitwise combination of:

User	Per User
Assembly	Per Assembly
Domain	Per Application Domain
Roaming	Move IS with Roaming Profile

(C) 2003, JL@Hisown.com - All Rights Reserved

IS

Isolated Storage: Example

```

class IsolatedStorageExample {
public static void Main() {
    IsolatedStorageFile isf = IsolatedStorageFile.GetStore(
        IsolatedStorageScope.Assembly|IsolatedStorageScope.User,
        null, null);

    IsolatedStorageFileStream isfs =
        new IsolatedStorageFileStream("file1",
                                       FileMode.OpenOrCreate,
                                       isf);
    StreamWriter sw = new StreamWriter(isfs);
    sw.WriteLine("This data is contained in IS");
    sw.Close();
    /* To Read: */
    isfs = new IsolatedStorageFileStream("file1", FileMode.Open, isf);
    StreamReader sr = new StreamReader(isfs);
    Console.WriteLine(sr.ReadToEnd());
    sr.Close();
}
}

```

(C) 2003, JL@Hisown.com - All Rights Reserved

Part II

Cryptography

“Go ert frzx etjjdds ax vgkex, xku xku acagz!”

(C) 2003, JL@Hisown.com - All Rights Reserved

Encryption

.Net Cryptographic Support

Built over CryptoAPI, providing support for:

- Encryption
- Digital Signatures
- Hashing
- Secure Random Number Generators

Implemented in [System.Security.Cryptography](#)

(C) 2003, JL @Hisown.com - All Rights Reserved

Encryption

.Net Encryption

Encryption: Turning plaintext into djqifsufyu.



Symmetric Cipher Suite Support


Alg	Key Size in Bits	Implementing Class
DES	64 (effective 56)	DESCryptoServiceProvider
3-DES	192 (effective 168)	TripleDESCryptoServiceProvider
RC2	40, 128	RC2CryptoServiceProvider
Rijndael	128 , 192 or 256	RijndaelManaged

(C) 2003, JL @Hisown.com - All Rights Reserved

Encryption


Symmetric Algorithms


System.Security.Cryptography. SymmetricAlgorithm (abstract class)	
May not be initialized - Created by a CryptoServiceProvider	
 Methods	
CreateDecryptor([Key], [IV])	Create a decryptor object with current Key and IV
CreateEncryptor([Key], [IV])	Create an encryptor object, with current Key and IV.
 Properties	
BlockSize	Get/Set the blocksize. May be one of LegalBlockSizes()
FeedbackSize	0..BlockSize. Specify feedback (OFB, CFB modes)
IV	The Initialization vector. Required for CBC.
Key	The most important aspect of the algorithm ☺
KeySize	Get/Set the keysize. May be one of LegalKeySizes()
Mode	CipherMode: One of ECB, CBC, CFB, OFB, or CTS.
Padding	PaddingMode: One of PKCS7, or Zeros


 2003, JL @Hisown.com - All Rights Reserved

Encryption

The CryptoStream

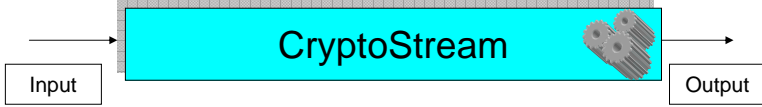


System.Security.Cryptography. CryptoStream	
(System.IO. Stream <i>stream</i> , ICryptoTransform <i>transform</i> , CryptoStreamMode <i>mode</i>) (any stream subclass) (any encryption algorithm) (read or write)	
 Methods	
Read/Write(buffer, offset, count)	Perform I/O on the cryptostream
ReadByte()/WriteByte()	Inherited from stream. Read/Write a single byte.
BeginRead/BeginWrite(..., callback, userdata)	As corresponding read/write, but with AsyncCallback and an opaque object – User supplied data.

 (C) 2003, JL @Hisown.com - All Rights Reserved

Encryption

Applying Encryption



Simply filter input stream through a cryptostream:

```

TripleDESCryptoServiceProvider DES3 = new TripleDESCryptoServiceProvider();
CryptoStream encStream = new CryptoStream(SomeFileStream,
                                           DES3.CreateEncryptor(tdesKey, IV),
                                           CryptoStreamMode.Write);

encStream.Write(buffer, offset, count);
encStream.read(buffer, offset, count);

```

(C) 2003, JL @Hisown.com - All Rights Reserved

.Net Digital Signatures

Digital Signatures: validating and authenticating messages

Supported: Digital Signature Algorithm (DSA)

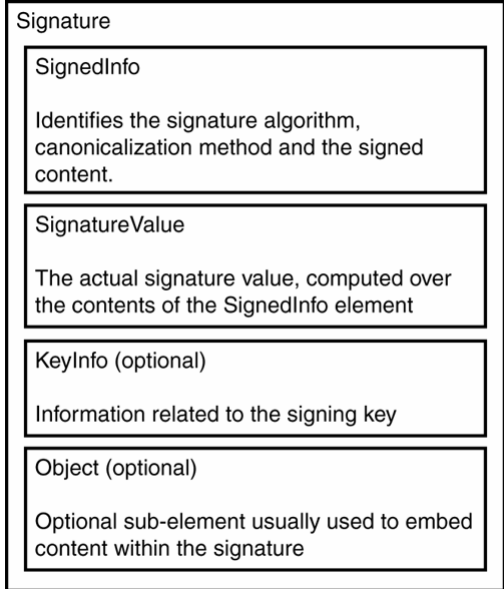
(Using SHA-1)

XML Digital Signatures (XMLDSIG)

(Using W3C signature standard)

(C) 2003, JL @Hisown.com - All Rights Reserved

XMLDSIG STRUCTURE

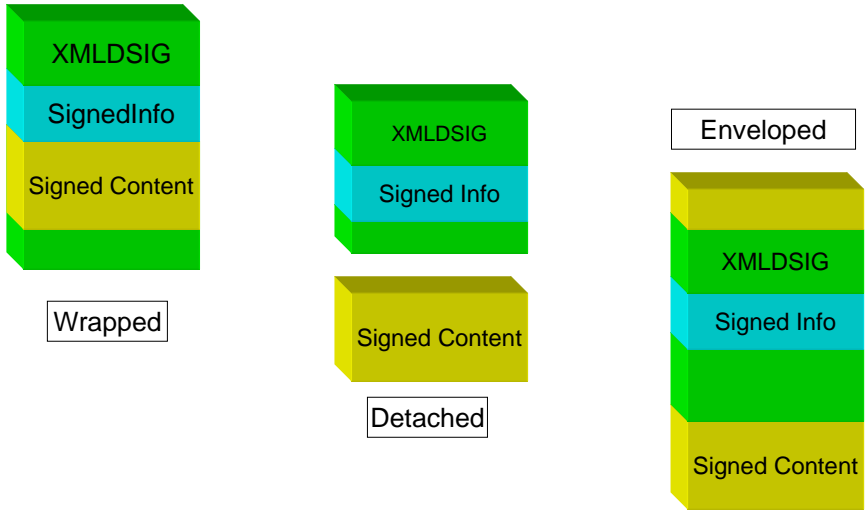


```

<element name="Signature"
  type="ds:SignatureType"/>
<complexType name="SignatureType">
  <sequence>
    <element ref="ds:SignedInfo"/>
    <element ref="ds:SignatureValue"/>
    <element ref="ds:KeyInfo" minOccurs="0"/>
    <element ref="ds:Object" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Id" type="ID" use="optional"/>
</complexType>

```

XMLDSIG Modes



(C) 2003, JL@Hisown.com - All Rights Reserved

Creating/Verifying XMLDSIG

```
public static XmlElement CreateDetachedSignature
(AsymmetricAlgorithm signingKey, String uri) {
    // Create the SignedXml message
    SignedXml signedXml = new SignedXml();
    signedXml.SigningKey = signingKey;
    // Create a Reference to point to the to-be-signed content
    Reference reference = new Reference(uri);
    // Add the Reference to the SignedXml message
    signedXml.AddReference(reference);
    // Compute the XML Digital Signature
    signedXml.ComputeSignature();
    // Return the XmlElement containing the XMLDSIG signature return
    signedXml.GetXml();
}
```

```
public static bool VerifySignature(XmlElement signatureElement) {
    SignedXml signedXml = new SignedXml();
    signedXml.LoadXml(signatureElement);
    return signedXml.CheckSignature();
}
```

(C) 2003, JL@Hisown.com - All Rights Reserved

.Net Hash Functions

Hash Function: Generate unique fixed length byte stream from arbitrary input.

Hash	Size in Bits	Class
MD5	128	MD5CryptoServiceProvider
SHA-1	160	SHA1CryptoServiceProvider
SHA256	256	SHA256Managed
SHA384	384	SHA384Managed
SHA512	512	SHA512Managed

(C) 2003, JL@Hisown.com - All Rights Reserved

.Net Hash Functions

Keyed Hash Function: Ditto, but with a key

Hash	Key Size in Bytes	Class
Keyed SHA1 (20 byte hash)	64	HMACSHA1
Triple Des CBC (8 Byte hash)	20	MACTripleDES

(C) 2003, JL @Hisown.com - All Rights Reserved

Random Number Generators

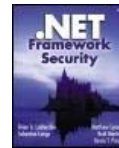
- Class RandomNumberGenerator
- Methods: GetBytes or GetNonZeroBytes

```
// Step 1:
RandomNumberGenerator prng =
    RandomNumberGenerator.Create();
// Create an array that will hold the random values
byte[] randomBytes = new byte[32];
// 256 bit key generation by filling with random bytes
prng.GetBytes(randomBytes);
```

(C) 2003, JL @Hisown.com - All Rights Reserved

References

- Foundstone assessment of .Net Security:
<http://www.microsoft.com/technet/itsolutions/net/evaluate/fsnetsec.asp>
- .Net Security Technical Article
<http://msdn.microsoft.com/vstudio/techinfo/articles/developerproductivity/frameworksec.asp>
- “.Net Framework Security”
ISBN: 0-672-32184-X, Addison Wesley



“Programming .Net Security”
ISBN: 0-596-00442-7 , O'Reilly

(C) 2003, JL @Hisown.com - All Rights Reserved

The End
(or perhaps, the beginning)

Questions? Comments?
JL@HisOwn.Com

(C) 2003, JL @Hisown.com - All Rights Reserved