



TestKingonline.com

**MCSE, CCNA, CCNP, OCP, CIW, JAVA, Sun Solaris, Checkpoint
World No 1 Cert Exams**

70-330

Congratulations!

You have purchased a TestKingonline. Study Guide.

This study guide is a complete collection of questions and answers that have been developed by our professional & certified team. You must study the contents of this guide properly in order to prepare for the actual certification test. The average time that we would suggest you for studying this study guide is approximately 10 to 20 hours and you will surely pass your exam. We guarantee it! GOOD LUCK!

DISCLAIMER

This study guide and/or material is not sponsored by, endorsed by or affiliated with Microsoft, Cisco, Oracle, Citrix, CIW, Checkpoint, Novell, Sun/Solaris, CWNA, LPI, ISC, etc. All trademarks are properties of their respective owners.

Guarantee

If you use this study guide correctly and still fail the exam, send a scanned copy of your official score notice at:
Sales@Testkingonline.com

We will gladly refund the cost of this study guide or give you an exchange of study guide of your choice of the Free.

This material is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this material, or any portion thereof, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Question: 1

You are an application developer for your company. You develop library assemblies that are called by your main applications. These library assemblies access confidential data in the applications. To ensure that this data is not accessed in an unauthorized and unsafe manner, users must not be allowed to call the library assemblies from their own applications. You apply a strong name to all assemblies to support versioning. You need to prevent users from writing managed applications that make calls to your library assemblies. You need to achieve this goal while minimizing the impact on response times for applications. What should you do?

- A. Use the internal access modifier to declare all classes and structs in each library.
- B. Use the protected internal access modifier to declare all classes and structs in each library.
- C. Add the following attribute to each class and struct in each library assembly.
<StrongNameIdentityPermission(SecurityAction.Demand, PublicKey:="002400..bda4")>
- D. Add the following attribute to each class and struct in each library assembly.
<StrongNameIdentityPermission(SecurityAction.LinkDemand, PublicKey:="002400..bda4")>

Answer: D

Question: 2

You are an application developer for Company.com. You are developing an application that can be extended by using custom components. The application uses reflection to dynamically load and invoke these custom components. In some cases, custom components will originate from a source that is not fully trusted, such as the Internet.

You need to programmatically restrict the code access security policy under which custom components run so that custom components do not run with an elevated permission grant.

What are two possible ways to achieve this goal? (Each correct answer presents a complete solution. Choose two)

- A. Create a new application domain and set the security policy level. Run custom components in this application domain.
- B. Use permission class operations to modify the security policy.
- C. Implement custom permission classes to protect custom component resources.
- D. Programmatically modify the machine-level security policy file after loading a custom component.

Answer: A, B

Question: 3

You are an application developer for your company. You are developing an application that salespeople in your company will use to process customer orders. This application includes a library assembly that implements a serviced component named Order. This serviced component adds roles named Manager and SalesPerson to the COM+ application that hosts it. To promote customer satisfaction, salespeople are allowed to apply discounts to orders if the order was erroneously delayed. However, only managers are allowed to apply discounts greater than 10 percent. The application includes the following method to apply the discount. Public Function ApplyDiscount(ByVal discountPct As Integer) As Boolean This method will return a value of False when the current user is not a member of the Manager role and the value of the discountPct parameter exceeds the maximum that other salespeople are allowed to apply. You need to add the code that will verify the role membership requirement when the value of discountPct is greater than 10. Which code segment should you use?

- A. If discountPct > 10 And _ Thread.CurrentPrincipal.IsInRole("Manager") = False Then Return FalseEnd If
- B. If discountPct > 10 Then Dim p As PrincipalPermission = New PrincipalPermission(Nothing, "Manager") If SecurityManager.IsGranted(p) = False Then Return False End IfEnd If
- C. If discountPct > 10 Then Dim p As PrincipalPermission = New PrincipalPermission(Nothing, "Manager") Try p.Demand() Catch e As SecurityException Return False End TryEnd If
- D. If discountPct > 10 And _ SecurityContext.CurrentCall.IsCallerInRole("Manager") _ = False Then Return FalseEnd If

Answer: D

Question: 4

You are an application developer for Company.com. You develop an application that receives data from a remote component. You are developing a method to detect any corrupted incoming data and log information to a file for analysis. You plan to use two functions. A function named CompanyData will be called by the remote component. The second function will be called by the local application to verify that the data was not corrupted during transmission. You need to ensure that corrupted data can be identified. Which code segment should you use?

- A. `Public Function CompanyData(ByVal Data As Byte()) As Byte() Dim Ms As New MemoryStream Ms.Write(Data, 0, Data.Length) Ms.Write(Data, 0, Data.Length) Return Ms.ToArray() End Function`
- B. `Public Function CompanyData(ByVal Data As Byte()) As Byte() Dim Md5 As MD5 = New MD5CryptoServiceProvider Dim Ms As New MemoryStream Ms.Write(Md5.ComputeHash(Data), 0, Md5.HashSize) Ms.Write(Data, 0, Data.Length) Return Ms.ToArray() End Function`
- C. `Public Function CompanyData(ByVal Data As Byte()) As Byte() Dim Des As DES = New DESCryptoServiceProvider Dim Ms As New MemoryStream Ms.Write(Des.Key, 0, Des.Key.Length) Ms.Write(Des.IV, 0, Des.IV.Length) Dim Cs As New CryptoStream(Ms, Des.CreateEncryptor(), CryptoStreamMode.Write) Cs.Write(Data, 0, Data.Length) Cs.FlushFinalBlock() Return Ms.ToArray() End Function`
- D. `Public Function CompanyData (ByVal Data As Byte()) As Byte() Dim Ms As New MemoryStream Dim Sw As New StreamWriter(Ms, Encoding.UTF8= Sw.Write(Encoding.UTF8.GetString(Data)) Return Ms.ToArray()`

Answer: B

Explanation

Hash functions map binary strings of an arbitrary length to small binary strings of a fixed length. A cryptographic hash function has the property that it is computationally infeasible to find two distinct inputs that hash to the same value; that is, hashes of two sets of data should match if the corresponding data also matches. Small changes to the data result in large unpredictable changes in the hash.

Example The following example computes the **MD5** hash for data and stores it in result. This example assumes that there is a predefined constant `DATA_SIZE`. `Dim data(DATA_SIZE) As Byte ' This is one implementation of the abstract class MD5. Dim md5 As New MD5CryptoServiceProvider() Dim result As Byte() = md5.ComputeHash(data)` It is easy to generate and compare hash values using the cryptographic resources contained in the **System.Security.Cryptography** namespace. Because all hash functions take input of type **Byte[]**, it might be necessary to convert the source into a byte array before it is hashed. To create a hash for a string value, follow these steps:

- 1 Open Visual Studio .NET.
- 2 Create a new Console Application in Visual Basic .NET. Visual Studio .NET creates a Module for you along with an empty `Main()` procedure.
- 3 Make sure that the project references the `System` and `System.Security` namespaces.
- 4 Use the `Imports` statement on the `System`, `System.Security`, `System.Security.Cryptography`, and `System.Text` namespaces so that you are not required to qualify declarations from these namespaces later in your code. These statements must be used prior to any other declarations.
- 5 `Imports System`
- 6 `Imports System.Security`
- 7 `Imports System.Security.Cryptography`
- 8 `Imports System.Text`
- 9 Declare a string variable to hold your source data, and two byte arrays (of undefined size) to hold the source bytes and the resulting hash value.
- 10 `Dim sSourceData As String`
- 11 `Dim tmpSource() As Byte`
- 12 `Dim tmpHash() As Byte`
- 13 Use the `GetBytes()` function, which is part of the `System.Text.ASCIIEncoding.ASCII` class, to convert your source string into an array of bytes (required as input to the hashing function).
- 14 `sSourceData = "MySourceData"`
- 15 'Create a byte array from source data.
- 16 `tmpSource = ASCIIEncoding.ASCII.GetBytes(sSourceData)`
- 17 Compute the MD5 hash for your source data by calling **ComputeHash** on an instance of the `MD5CryptoServiceProvider` class. Note that to compute another hash value, you will need to create another instance of the class.

```

18 'Compute hash based on source data.
19 tmpHash = New MD5CryptoServiceProvider().ComputeHash(tmpSource)
20 The tmpHash byte array now holds the computed hash value (128-bit value=16 bytes) for your source data. It is often
    useful to display or store a value like this as a hexadecimal string, which the following code accomplishes:
21 Console.WriteLine(ByteArrayToString(tmpHash))
22
23 Private Function ByteArrayToString(ByVal arrInput() As Byte) As String
24 Dim i As Integer
25 Dim sOutput As New StringBuilder(arrInput.Length)
26 For i = 0 To arrInput.Length - 1
27 sOutput.Append(arrInput(i).ToString("X2"))
28 Next
29 Return sOutput.ToString()
30 End Function
31 Save and then run your code to see the resulting hexadecimal string for the source value.

```

Question: 5

You are an application developer for your company, which is named Company.com. You are developing an ASP.NET Web application that users in the accounting department will use to process payroll reports and view payroll reports. The application will use Integrated Windows authentication to authenticate all users. Because payroll data is confidential only users in the accounting department will be granted access to the application. All employees in the accounting department belong to a specific Active Directory group. However, users in the IT department can add themselves to various Active Directory groups in order to troubleshoot resource access problems. These IT department users must not be granted access to the ASP.NET Web application. The following rules can be used to distinguish between users in the accounting department and users in the IT department:

- All users in the accounting department are members of a group named Company\Accounting.
- Some users in the IT department are members of the Company\Accounting group.
- All users in the IT department are members of a group named Company\Domain Admin.
- No users in the accounting department are members of the Company\Domain Admin group.

You need to configure URL authorization for the application by adding an <authorization> element to the Web.config file in the application root. Which element should you use?

- A. <authorization> <deny roles="Company\Domain Admin"/> <allow roles="Company\Accounting"/> <deny users="*/>
</authorization>
- B. <authorization> <allow roles="Company\Accounting"/> <deny roles="Company\Domain Admin"/> <deny users="*/>
</authorization>
- C. <authorization> <deny roles="Domain Admin"/> <allow
roles="Accounting"/> <deny users="*/> </authorization>
- D. <authorization> <allow roles="Accounting"/> <deny
roles="Domain Admin"/>
<deny users="*/> </authorization>

Answer: A

Explanation

<authorization> Element Configures ASP.NET authorization support. The <authorization> tag controls client access to URL resources. This element can be declared at any level (machine, site, application, subdirectory, or page). <configuration>
<system.web>
<authorization><authorization><allow users="comma-separated list of users" roles="comma-separated list of roles" verbs="comma-separated list of verbs"/> <deny users="comma-separated list of users" roles="comma-separated list of roles" verbs="comma-separated list of verbs"/> </authorization> Subtag Description <allow> Allows access to a resource based on the following: **users:** A

comma-separated list of user names that are granted access to the resource. A question mark (?) allows anonymous users; an asterisk (*) allows all users. **roles**: A comma-separated list of roles that are granted access to the resource. **verbs**: A comma-separated list of HTTP transmission methods that are granted access to the resource. Verbs registered to ASP.NET are GET, HEAD, POST, and DEBUG. <deny> Denies access to a resource based on the following: **users**: A comma-separated list of user names that are denied access to the resource. A question mark (?) indicates that anonymous users are denied access; an asterisk (*) indicates that all users are denied access. **roles**: A comma-separated list of roles that are denied access to the resource. **verbs**: A comma-separated list of HTTP transmission methods that are denied access to the resource. Verbs registered to ASP.NET are GET, HEAD, POST, and DEBUG.

Remarks

At run time, the authorization module iterates through the <allow> and <deny> tags until it finds the first access rule that fits a particular user. It then grants or denies access to a URL resource depending on whether the first access rule found is an <allow> or a <deny> rule. The default authorization rule in the Machine.config file is <allow users="*" /> so, by default, access is allowed unless configured otherwise.

Example

The following example allows access to all members of the Admins role and denies access to all users. <configuration>
<system.web><authorization><allow roles="Admins" /> <deny users="*" /></authorization> </system.web></configuration>

Question: 6

You are an application developer for Company.com. Your team is developing a Windows Forms application. Users will have access to different functionality depending on their roles in Company. The application includes the following method.

```
Private Shared Function AuthenticateUser (ByVal user As String, _
ByVal password As String, ByRef roles As String()) As
Boolean
```

This method authenticates the user against a third-party data store. When authentication is successfully, this method returns a value of True, and the string array named roles is updated to contain the user's roles. You need to write the code that associates an authenticated user and the user's roles with the current security context.

Which code segment should you use?

- A. ' p is initialized above as a PrincipalPermission If AuthenticateUser (name, password, roles) = True Then Dim r As String For Each r In Roles Dim ppTemp As PrincipalPermission = New PrincipalPermission(name, r) p.Union(ppTemp) Next End If p.IsUnrestricted()
- B. ' p is initialized above as a PrincipalPermission If AuthenticateUser (name, password, roles) = True Then Dim r As String For Each r In roles Dim ppTemp As PrincipalPermission = New PrincipalPermission(name, r) Next End If p.IsUnrestricted()
- C. If AuthenticateUser(name, password, roles) = True Then Dim r As String For Each r In roles Thread.CurrentPrincipal.IsInRole(r) Next End If
- D. If AuthenticateUser(name, password, roles) = True Then Thread.CurrentPrincipal = New GenericPrincipal(New GenericIdentity(name), roles) End If

Answer: D

Explanation Difference Between Declarative and Imperative Security

There are two main differences between the use of declarative security and imperative security. In declarative security, the roles are essentially hard coded at design time, while in imperative security, these can be read from an external source such as a database or a config file. While config files can be used for prototypes or very simple applications, databases should be the repository of choice for roles. Further, with declarative security, the granularity of the access check is a method, while with imperative security, the granularity is controlled by the developer. The following code illustrates how a caller of such a component can communicate the roles it belongs to. This is shown below: private void SetupPrincipal () { string IUserName = cbUserName.SelectedItem.ToString (); GenericIdentity IIdentity = new GenericIdentity (IUserName); string[] IDMRoles = {"DistrictManagers"}; string[] IRMRoles = {"RegionalManagers"}; GenericPrincipal IPPrincipal;if (IUserName.Equals ("Alex DM")) IPPrincipal = new GenericPrincipal(IIdentity, IDMRoles);if (IUserName.Equals ("Tony RM")) IPPrincipal = new GenericPrincipal(IIdentity, IRMRoles);

Thread.CurrentPrincipal = IPrincipal; } We first create an identity for the user, and then, based on the user name, associate the user name and a role to create a principal. This principal is then attached to the current thread object so that this can be accessed by all downstream components.

Creating GenericPrincipal and GenericIdentity Objects

You can use the GenericIdentity class in conjunction with the GenericPrincipal class to create an authorization scheme that exists independent of a Windows NT or Windows 2000 domain. Perform the following tasks to create an instance of the **GenericPrincipal** class.

1 Create a new instance of the identity class and initialize it with the name you want it to hold. The following code creates a new **GenericIdentity** object and initializes it with the name MyUser.

2 [C#]

3 GenericIdentity MyIdentity = new GenericIdentity("MyUser");

4 Dim MyIdentity As New GenericIdentity("MyUser")

5 Next, create a new instance of the **GenericPrincipal** class and initialize it with the previously created GenericIdentity object and an array of strings that represent the roles that you want associated with this principal. The following code example specifies an array of strings that represent an administrator role and a user role. The GenericPrincipal is then initialized with the previous GenericIdentity and the string array.

6 [C#]

7 String[] MyStringArray = {"Manager", "Teller"};

8 GenericPrincipal MyPrincipal = new GenericPrincipal(MyIdentity, MyStringArray);

9

10 Dim MyStringArray As String() = {"Manager", "Teller"} Dim MyPrincipal As New GenericPrincipal(MyIdentity, MyStringArray)

11 Finally, use the following code to attach the principal to the current thread. This is valuable in situation where the principal must be validated several times, it must be validated by other code running in your application, or it must be validated by a PrincipalPermission object. You can still perform role-based validation on the principal object without attaching it to the thread. For more information, see Replacing a Principal Object.

12 [C#]

13 Thread.CurrentPrincipal = MyPrincipal;

14 Thread.CurrentPrincipal = MyPrincipal The following code example demonstrates how to create an instance of a GenericPrincipal and a GenericIdentity. This code displays the values of these classes to the console. [C#] using System; using System.Security.Principal; using System.Threading; public class Class1 { public static int Main(string[] args) { //Create generic identity. GenericIdentity MyIdentity = new GenericIdentity("MyIdentity"); //Create generic principal. String[] MyStringArray = {"Manager", "Teller"}; GenericPrincipal MyPrincipal = new GenericPrincipal(MyIdentity, MyStringArray); //Attach the principal to the current thread. //This is not required unless repeated validation must occur, //other code in your application must validate, or the

// PrincipalPermission object is used. Thread.CurrentPrincipal = MyPrincipal; //Print values to the console. String Name = MyPrincipal.Identity.Name; bool Auth = MyPrincipal.Identity.IsAuthenticated; bool IsInRole = MyPrincipal.IsInRole("Manager"); Console.WriteLine("The Name is: {0}", Name); Console.WriteLine("The IsAuthenticated is: {0}", Auth); Console.WriteLine("Is this a Manager? {0}", IsInRole);return 0; } } Imports System Imports System.Security.Principal Imports System.ThreadingPublic Class Class1 Public Shared Sub Main() 'Create generic identity. Dim MyIdentity As New GenericIdentity("MyIdentity") 'Create generic principal. Dim MyStringArray As String() = {"Manager", "Teller"}Dim MyPrincipal As New GenericPrincipal(MyIdentity, MyStringArray) 'Attach the principal to the current thread. 'This is not required unless repeated validation must occur, 'other code in your application must validate, or the ' PrincipalPermission object is used. Thread.CurrentPrincipal = MyPrincipal 'Print values to the console.Dim Name As String = MyPrincipal.Identity.Name Dim Auth As Boolean = MyPrincipal.Identity.IsAuthenticated Dim IsInRole As Boolean = MyPrincipal.IsInRole("Manager")Console.WriteLine("The Name is: {0}", Name) Console.WriteLine("The IsAuthenticated is: {0}", Auth) Console.WriteLine("Is this a Manager? {0}", IsInRole) End Sub End Class When executed, the application displays output similar to the following. The Name is: MyIdentity The IsAuthenticated is: TrueIs this a Manager? True

Question: 7

You are an application developer for your company. You are developing a three-tier Windows Forms application that will be used to manage confidential records. The business layer includes a remote object that is installed on an application server. The remote object is hosted in ASP.NET on the application server. IIS is configured to use Integrated Windows authentication, and ASP.NET is configured to use Windows authentication. All client computers and servers on the network support Kerberos authentication. The Windows Forms application communicates with the remote object by using a remoting proxy named myProxy. The remote object accesses a Microsoft SQL Server database. Permissions to database objects are granted based on the identity of the user. The remote object needs to run under the security context of the user. You need to write the code in the Windows Forms application that will configure the remoting proxy to have the credentials to use for authentication. Which code segment should you use?

- A. Dim channelProperties As IDictionarychannelProperties = ChannelServices.GetChannelSinkProperties(myProxy)channelProperties("credentials") = CredentialCache.DefaultCredentials
- B. Dim channelProperties As IDictionaryDim cred As NetworkCredential = New NetworkCredential(_userName, _psswd)channelProperties = ChannelServices.GetChannelSinkProperties(myProxy)channelProperties("credentials") = cred
- C. Dim channelProperties As IDictionarychannelProperties = ChannelServices.GetChannelSinkProperties(myProxy)channelProperties("credentials") = Thread.CurrentPrincipal
- D. Dim channelProperties As IDictionarychannelProperties = ChannelServices.GetChannelSinkProperties(myProxy)channelProperties("credentials") = Thread.CurrentPrincipal.Identity

Answer: A

Explanation Configure Client Credentials

To successfully communicate with a remote component that is configured for Windows authentication, the client must configure the remoting proxy with the credentials to use for authentication. Failure to do so results in an access denied error. You can configure the use of default credentials to use the client's current thread or process token, or you can set explicit credentials.

Using Default Credentials

To use the client's process token (or thread token if the client thread is currently impersonating), set the **useDefaultCredentials** property of the client proxy to **true**. This results in the use of **CredentialsCache.DefaultCredentials** when the client receives an authentication challenge from the server. You can configure the proxy either by using the configuration file or programmatically in code. To configure the proxy externally, use the following element in the client configuration file: <channel ref="http client" useDefaultCredentials="true" /> To set default credentials programmatically, use the following code: IDictionary channelProperties; channelProperties = ChannelServices.GetChannelSinkProperties(proxy); channelProperties ["credentials"] = CredentialCache.DefaultCredentials; If you use default credentials in an ASP.NET client application that is configured for impersonation, the thread level impersonation token is used. This requires Kerberos delegation.

Using Alternate Credentials

To use a specific set of credentials for authentication when you call a remote object, disable the use of default credentials within the configuration file by using the following setting. <channel ref="http" useDefaultCredentials="false" /> **Note** Programmatic settings always override the settings in the configuration file. Then, use the following code to configure the proxy to use specific credentials: IDictionary channelProperties = ChannelServices.GetChannelSinkProperties(proxy); NetworkCredential credentials; credentials = new NetworkCredential("username", "password", "domain"); ObjRef objectReference = RemotingServices.Marshal(proxy); Uri objectUri = new Uri(objectReference.URI); CredentialCache credCache = new CredentialCache(); // Substitute "authenticationType" with "Negotiate", "Basic", "Digest", // "Kerberos" or "NTLM" credCache.Add(objectUri, "authenticationType", credentials); channelProperties["credentials"] = credCache; channelProperties["preauthenticate"] = true;

Question: 8

You are an application developer for Company.com. You develop an ASP.NET Web application for Company's intranet. The application accesses data that is stored in a Microsoft SQL Server database. The application authenticates users by using Windows authentication, and it has impersonation enabled. You configure database object permissions based on the identity of the user of the application. You need to provide the user's identity to the SQL Server database. What should you do?

- A. Connect to the database by using the following connection string "Persists Security

Info=False;Integrated Security=SSPI; database=ApplicationDB;server=DataServer;"

- B. Connect to the database by using the following connection string "User ID=ASPNET;Persist Security Info=False;Integrated Security=False; database=ApplicationDB;server=DataServer;"
- C. Develop a serviced component that wraps all database operations. Use COM+ role-based security to restrict access to database operations based on user identity.
- D. Disable impersonation.

Answer: A

Explanation

We need to configure the following four different areas to access Windows integrated security:

- 1 SQL Server
- 2 IIS Web Server
- 3 ASP.Net web application
- 4 ConnectionString SQL Server be running on same IIS machine. If both are on different machines, we should go for an alternative security model such as Forms authentication, or Kerberos delegation would need to be used. The access users must be in the same domain where the Web server is running. Configuring SQL Server To configure SQL Server for Windows integrated security:

- 1 From the Windows Start menu, choose Microsoft SQL Server, and then choose Enterprise Manager.
- 2 Open the node for the server and expand the node for the database you want to give users permissions for.
- 3 Right-click the **Users** node and choose **New Database User**.
- 4 In the **Database User Properties** dialog box, enter *domain\username* in the **Login name** box, and then click **OK**.

Alternatively, configure the SQL Server to allow all domain users to access the database. Configuring IIS You need to configure your application in IIS to turn off anonymous access and turn on Windows authentication. To configure IIS for Windows integrated security:

- 1 In Windows, open the Internet Information Services administration tool.
- 2 Open the node for your server, and then open nodes until you find the node for your application, typically under Default Web Site.
- 3 Right-click your application and choose Properties.
- 4 In the Directory Security tab, click Edit.
- 5 In the Authentication Methods dialog box, clear the Anonymous Access box and make sure Integrated Windows authentication is checked.
- 6 Click OK to close all the dialog boxes. Configuring the ASP.NET Web Application

In the application configuration file (Web.config), you establish the authentication mode that your application uses and establish that the application will impersonate the user's credentials—that is, that it will run as that user. To configure Web.config to allow Windows integrated security: Open the Web.config file for your application and add the following elements to it: <authentication mode="Windows" /> <identity impersonate="true"/> The <authentication> element might already be there. Creating Connection Strings When you create a connection string to access SQL Server, you must include attributes that tell SQL Server that you are using Windows integrated security. To configure connection strings for Windows integrated security: In any connection string for SQL Server, include the Trusted_Connection=Yes attribute and remove the username and password attributes. The following shows a typical connection string configured for Windows integrated security: "data source=Sql01;initial catalog=Northwind; integrated security=SSPI;persist security info=False; Trusted_Connection=Yes." Sample C# code for connecting SQL server from ASP.Net application using windows authentication: private void DataBind() { SqlConnection = SqlConnection("data source=bondugula; initial catalog=Northwind; integrated security=SSPI; persist security info=False;Trusted_Connection=Yes"); SqlConnection.Open(); SqlDataAdapter = new SqlDataAdapter("SELECT EmployeeID, FirstName, LastName, Title FROM Employees", SqlConnection);

```
dataSet = new DataSet(); sqlDataAdapter.Fill(dataSet, "Employees"); DataGrid1.DataSource =
dataSet.Tables["Employees"].DefaultView; DataGrid1.DataBind(); } Important settings in the web.config file are as follows:
<system.web><authentication mode = "Windows"/> <identity impersonate="true"/> <authorization><allow users = "*" />
</authorization> <!--other settings--> </system.web>
```

Question: 9

You are an application developer for your company. You are developing an application that receives signed data. The data is signed by using the RSA encryption algorithm and the SHA1 hash algorithm. You need to write a function that will verify signatures by using RSA public credentials. Which code segment should you use?

- A. Public Function VerifySignature(ByVal Data As Byte(), ByVal Signature As Byte(), _ ByVal RsaKey As RSAParameters) As Boolean Dim RSA As New RSACryptoServiceProvider RSA.ImportParameters(RsaKey) Dim MySig As Byte() = RSA.SignData(Data, "SHA1") Dim i As Integer For i = 0 To MySig.Length - 1 If i >= Signature.Length Or Signature(i) <> MySig(i) Then Return False End If Next Return True End Function
- B. Public Function VerifySignature(ByVal Data() As Byte, ByVal Signature As Byte(), _ ByVal RsaKey As RSAParameters) As Boolean Dim RSA As New RSACryptoServiceProvider RSA.ImportParameters(RsaKey) Return RSA.VerifyData(Data, "SHA1", Signature) End Function
- C. Public Function VerifySignature(ByVal Data As Byte(), ByVal Signature As Byte(), _ ByVal RsaKey As RSAParameters) As Boolean Dim RSA As New RSACryptoServiceProvider RSA.ImportParameters(RsaKey) Dim MySig As Byte() = RSA.Decrypt(Data, False) Dim i As Integer For i = 0 To MySig.Length - 1 If i >= Signature.Length Or Signature(i) <> MySig(i) Then Return False End If Next Return True End Function
- D. Public Function VerifySignature(ByVal Data As Byte(), ByVal Signature As Byte(), _ ByVal RsaKey As RSAParameters) As Boolean Dim RSA As New RSACryptoServiceProvider RSA.ImportParameters(RsaKey) Dim shaOID As String = CryptoConfig.MapNameToOID("SHA1") Return RSA.VerifyHash(Data, shaOID, Signature) End Function

Answer: B

Explanation

Verifying Signatures In order to verify that data was signed by a particular party, you must have the following information:

- The public key of the party that signed the data.

- The digital signature.

- The data that was signed.

The hash algorithm used by the signer. To verify a signature signed by the RSAPKCS1SignatureFormatter class, use the RSAPKCS1SignatureDeformatter class. The RSAPKCS1SignatureDeformatter class must be supplied the public key of the signer. You will need the values of the modulus and the exponent to specify the public key. (The party that generated the public/private key pair should provide these values.) First create an RSACryptoServiceProvider object to hold the public key that will verify the signature, and then initialize an RSAParameters structure to the modulus and exponent values that specify the public key. The following code shows the creation of an RSAParameters structure. The Modulus property is set to the value of a byte array called ModulusData and the Exponent property is set to the value of a byte array called ExponentData. Dim RSAKeyInfo As RSAParameters RSAKeyInfo.Modulus = ModulusData

RSAKeyInfo.Exponent = ExponentData After you have created the RSAParameters object, you can initialize a new instance of the RSACryptoServiceProvider class to the values specified in RSAParameters. The RSACryptoServiceProvider is, in turn, passed to the constructor of an RSAPKCS1SignatureDeformatter to transfer the key. The following example illustrates this process. In this example, HashValue and SignedHashValue are arrays of bytes provided by a remote party. The remote party has signed the HashValue using the SHA1 algorithm, producing the digital signature SignedHashValue. The RSAPKCS1SignatureDeformatter.VerifySignature method verifies that the digital signature is valid and was used to sign the HashValue. Dim RSA As New RSACryptoServiceProvider() RSA.ImportParameters(RSAKeyInfo) Dim RSADeformatter As New RSAPKCS1SignatureDeformatter(RSA) RSADeformatter.SetHashAlgorithm("SHA1") If RSADeformatter.VerifySignature(HashValue, SignedHashValue) Then Console.WriteLine("The signature is valid.") Else Console.WriteLine("The signature is not valid.") End If The above code fragment will display "The signature is valid" if the signature is valid and "The signature is not valid" if it is not. RSACryptoServiceProvider.VerifyData Method Verifies the specified signature data by comparing it to the signature computed for the specified data. [Visual Basic] Public Function VerifyData(_ ByVal buffer() As Byte, _ ByVal hash As Object, _ ByVal signature() As Byte _) As Boolean

Parametersbuffer The data that was signed. halg The name of the hash algorithm used to create the hash value of the data.
signature The signature data to be verified.

Return Value

true if the signature verifies as valid; otherwise, false.

Remarks This method verifies the RSA digital signature produced by SignData. The halg parameter can accept a String, a HashAlgorithm, or a Type.

RSACryptoServiceProvider.VerifyHash Method Verifies the specified signature data by comparing it to the signature computed for the specifiedhash value. [Visual Basic]Public Function VerifyHash(_ ByVal rgbHash() As Byte, _
ByVal str As String, _ ByVal rgbSignature() As Byte _) As Boolean

Parametersbuffer The data that was signed. halg The name of the hash algorithm used to create the hash value of the data. signature The signature data to be verified.

Return Value

true if the signature verifies as valid; otherwise, false.

Remarks This method verifies the RSA digital signature produced by SignData. The halg parameter can accept a String, a HashAlgorithm, or a Type.

Question: 10

You are an application developer for your company. You are developing an application that reads the USERNAME environment variable and executes code in an unmanaged DLL. The design document specifies that the application must display a custom message when the code access security policy restricts access to required resources. You need to write the code segment that will ascertain whether your application is permitted to access unmanaged code and the USERNAME environment variable. Your solution must allow the application to display the custom message when the application is being loaded. Which code segment should you use?

- A. Try Dim ep As EnvironmentPermission = New _ EnvironmentPermission(EnvironmentPermissionAccess.Read, "USERNAME")
Dim sp As SecurityPermission = New _ SecurityPermission(SecurityPermissionFlag.UnmanagedCode) ep.Demand()
sp.Demand()Catch ex As SecurityException '...End Try
- B. Dim ep As EnvironmentPermission = New _ EnvironmentPermission(EnvironmentPermissionAccess.Read, "USERNAME")Dim sp
As SecurityPermission = New _ SecurityPermission(SecurityPermissionFlag.UnmanagedCode)If Not (ep.IsUnrestricted() And
sp.IsUnrestricted()) Then ' ... End If
- C. <EnvironmentPermission(SecurityAction.Demand, Read:="USERNAME"), _ SecurityPermission(SecurityAction.Demand,
UnmanagedCode:=True)> _Sub Main() ' ... End Sub
- D. <Assembly: EnvironmentPermission(SecurityAction.RequestMinimum, Read:="USERNAME")><Assembly:
SecurityPermission(SecurityAction.RequestMinimum, UnmanagedCode:=True)>

Answer: B

Question: 11

You are an application developer for your company. You are developing a Windows-based application that stores user configuration information for the application. The information is stored in a file named Application.config. You need to ensure that only users in the Administrators group can make changes to the configuration of the application. What should you do?

- A. Encrypt Application.config by using the Administrators group's private key. Decrypt the file prior to reading its data from the application.
- B. Set a discretionary access control list (DACL) on Application.config that grants the Administrators group Read permission and Write permission, but grants other users only Read permission.

- C. Add the following code segment to the assembly settings of the application. <Assembly: FileIOPermission(SecurityAction.RequestMinimum, _ Read:="C:\Program Files\MyApp\Application.config")>Add the following code segment to the start of the Main() function in the application code. Dim Wi As WindowsIdentityWi = WindowsIdentity.GetCurrent()Dim Wp As New WindowsPrincipal(Wi)If Wp.IsInRole("BUILTIN\Administrators") Then Dim Fip As New FileIOPermission _ (FileIOPermissionAccess.AllAccess,"Application.config") Fip.Assert()End If
- D. Add the following code segment at each point that Application.config is opened. Dim Wi As WindowsIdentityWi = WindowsIdentity.GetCurrent()Dim Wp As New WindowsPrincipal(Wi)If Not Wp.IsInRole("BUILTIN\Administrators") Then Dim Fip As New FileIOPermission _ (FileIOPermissionAccess.Read,"Application.config") Fip.PermitOnly()End If

Answer: C

Question: 12

You are an application developer for Company.com. You create an ASP.NET Web application that all authenticated network users will access. The authentication mode in the Web.config file is currently set to None. Due to recent security threats, the network administrator requires that all connections to the application's Web server use the network credentials of the authenticated user. You need to configure the application to use the network credentials of the authenticated user as HttpContext.Current.User. Which action or actions should you perform? (Choose all that apply)

- A. Ask the network administrator to configure the IIS directory security to Anonymous authentication.
- B. Ask the network administrator to configure the IIS directory security to Integrated Windows authentication.
- C. Set the authentication mode in the Web.config file to **Forms**.
- D. Set the authentication mode in the Web.config file to **Windows**.
- E. Set the impersonation attribute of the identity element in the Web.config file to **true**.

Answer: D, E

Explanation

Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity. Once an identity has been authenticated, the authorization process determines whether that identity has access to a given resource. Integrated Windows Authentication Integrated Windows authentication uses Windows logon credentials to authenticate users. Rather than prompt a user for a user name and password and transmit them over HTTP, a browser asked to identify the user through integrated Windows authentication carries on a conversation with the Web server and identifies the user using that person's login identity on the client. ASP.NET implements authentication through authentication providers, the code modules that contain the code necessary to authenticate the requestor's credentials. ASP.NET supports the authentication providers described in the following table.

ASP.NET authentication provider	Description
Forms authentication	A system by which unauthenticated requests are redirected to an HTML form using HTTP client-side redirection. The user provides credentials and submits the form. If the application authenticates the request, the system issues a cookie that contains the credentials or a key for reacquiring the identity. Subsequent requests are issued with the cookie in the request headers; they are authenticated and authorized by an ASP.NET event handler using whatever validation method the application developer specifies.
Passport authentication	Centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites.

Windows authentication	ASP.NET uses Windows authentication in conjunction with Microsoft Internet Information Services (IIS) authentication. Authentication is performed by IIS in one of three ways: basic, digest, or Integrated Windows Authentication. When IIS authentication is complete, ASP.NET uses the authenticated identity to authorize access.
-------------------------------	---

To enable an authentication provider for an ASP.NET application, you only need to create an entry for the application configuration file as follows. // Web.config file <authentication mode= "[Windows|Forms|Passport|None]"/> The mode is set to one of the authentication modes: Windows, Forms, Passport, or None. The default is Windows. If the mode is None, ASP.NET does not apply any additional authentication to the request - this can be useful when you want to implement a custom authentication scheme, or if you are solely using anonymous authentication and want the highest possible level of performance. The authentication mode cannot be set at a level below the application root directory. As is the case with other ASP.NET modules, subdirectories in the URL space inherit authentication modules unless explicitly overridden.

The WindowsAuthenticationModule provider relies on Microsoft Internet Information Services (IIS) to provide authenticated users, using any of the mechanisms that IIS supports. If you want to implement site security with a minimum of ASP.NET coding, this is the provider configuration you should use. The provider module constructs a WindowsIdentity object. The default implementation constructs a WindowsPrincipal object and attaches it to the application context. The WindowsPrincipal object maps identities to Windows groups.

If you use IIS authentication, the provider module uses the authenticated identity passed in from IIS. IIS authenticates the identity using basic, digest, or Integrated Windows authentication, or some combination of them. You can use impersonation and NTFS ACL permissions to restrict or allow access to protected resources. An important reason to use the WindowsAuthenticationModule provider is to implement an impersonation scheme that can use any of the authentication methods that might have already been performed by IIS before passing the request to the ASP.NET application. To do this, set the authentication mode to Windows, and confirm that the impersonate element is set to true, as shown in the following example: <authentication mode="Windows"/> <identity impersonate="true"/> Please note that configuring an ASP.NET application has no effect on the IIS Directory Security settings.

The systems are completely independent and are applied in sequence. In addition to selecting an authentication mode for an ASP.NET application, it is also important to configure IIS authentication appropriately. Next, you must set the NTFS ACLs to allow access only to the proper identities. If you want to enable impersonation for only a short time during request processing, you can do it by using an impersonation context and WindowsIdentity.Impersonate. First, set the impersonate element to false, and then set up a context using the WindowsIdentity.Impersonate method, as shown in the following example. Dim context As WindowsImpersonationContext = _ WindowsIdentity.Impersonate(impersonateToken) ' Perform some action. context.Undo() Notice that you can use context.Undo for identity reversion. As mentioned earlier, you can implement a custom Windows authorization scheme by using a WindowsAuthentication_OnAuthenticate event handler to create a WindowsPrincipal or a GenericPrincipal object from a WindowsIdentity object. You can then use one of the new objects to implement your own custom authentication scheme. The WindowsPrincipal object maps identities to Windows groups. The default implementation constructs a WindowsPrincipal object and attaches it to the application context.

Question: 13

You are an application developer for Company.com. You develop a Windows Forms application. You want your application to use a class library that was developed by another developer. You run the Permissions View tool on the class library and receive the following output.

Microsoft (R) .NET Framework Permission Request Viewer. Version 1.1.4322.573 Copyright (C) Microsoft Corporation 1998-2002. All rights reserved. minimal permission set: <PermissionSet class="System.Security.PermissionSet" version="1"><IPermission class="System.Security.Permissions.FileIOPermission,mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Write="C:\SecureFile.txt"/> <IPermission class="System.Security.Permissions.ReflectionPermission, mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Flags="ReflectionEmit"/> <IPermission class="System.Security.Permission.SecurityPermission, mscorlib, Version=1.0.5000.0, Culture=netrual, PublicKeyToken=b77a5c561934e089" version="1" Flags="SerializationFormatter"/> </PermissionSet>optional permission set: <PermissionSet class="System.Security.PermissionSet" version="1" Unrestricted="true"/> refused permission set: Not specified You need to add the correct attributes to the Windows Forms application code before the call to the class library. Which code segment should you use?

- A. <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, _ ReflectionEmit:=False), _ Assembly: SecurityPermission(SecurityAction.RequestOptional, _ SerializationFormatter:=False), _ Assembly: FileIOPermissionAttribute(SecurityAction.RequestRefuse, _ Write:="C:\SecureFile.txt"), _ Assembly: PermissionSetAttribute(SecurityAction.RequestOptional, Unrestricted:=True)>
- B. <Assembly: ReflectionPermission(SecurityAction.RequestOptional, _ ReflectionEmit:True), _ Assembly: SecurityPermission(SecurityAction.RequestMinimum, _ SerializationFormatter:=True), _ Assembly: FileIOPermissionAttribute(SecurityAction.RequestOptional, _ Write:="C:\SecureFile.txt"), _ Assembly: PermissionSetAttribute(SecurityAction.RequestMinimum, Unrestricted:=True)>
- C. <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, _ ReflectionEmit:=False), _ Assembly: SecurityPermission(SecurityAction.RequestMinimum, _ SerializationFormatter:=False), _ Assembly: FileIOPermissionAttribute(SecurityAction.RequestOptional, _ Write:="C:\SecureFile.txt"), _ Assembly: PermissionSetAttribute(SecurityAction.RequestMinimum, Unrestricted:=True)>
- D. <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, _ ReflectionEmit:=True), _ Assembly: SecurityPermission(SecurityAction.RequestMinimum, _ SerializationFormatter:=True), _ Assembly: FileIOPermissionAttribute(SecurityAction.RequestMinimum, _

Answer: D

Explanation

The SDK provides a tool called PERMVIEW that is useful for verifying that your permission requests are correct. Running PERMVIEW on a compiled assembly will read the permission requests out of the assembly's manifest and display them as shown

below. C:\>permview filemove.exe Microsoft (R) .NET Framework Permission Request Viewer. Version 1.0.XXXX.0 Copyright (C) Microsoft Corp. 1998-2001 minimal permission set: <PermissionSet class="System.Security.PermissionSet" version="1"><IPermission class="System.Security.Permissions.FileIOPermission" version="1" Unrestricted="true"/></PermissionSet>optional permission set: <PermissionSet class="System.Security.PermissionSet" version="1" Unrestricted="true"/>refused permission set: Not specified The Permissions View tool is used to view the minimal, optional, and refused permission sets requested by an assembly. Optionally, you can use Permview.exe to view all declarative security used by an assembly. permview [/output *filename*] [/decl] *manifestfile*

Argument	Description
<i>manifestfile</i>	The file that contains the assembly's manifest. The manifest can be either a
	standalone file or it can be incorporated in a portable executable (PE) file. The extension for this file will usually be .exe or .dll, but it could also be .scr, or .ocx.
Option	Description
<i>/decl</i>	Displays all declarative security at the assembly, class, and method level for the assembly specified by <i>manifestfile</i> . This includes permission requests as well as demands, asserts, and all other security actions that can be applied declaratively. It does not refer to other assemblies linked to the specified assembly.
<i>/h[elp]</i>	Displays command syntax and options for the tool.
<i>/output</i>	<i>filename</i> Writes the output to the specified file. The default is to display the output to the console.
<i>/?</i>	Displays command syntax and options for the tool.

Remarks
Developers can use Permview.exe to verify that they have applied permission requests correctly to their code. Additionally, users can run Permview.exe to determine the permissions an assembly requires to execute. For example, if you run a managed executable and get the error, "System.Security.Policy.PolicyException: Failed to acquire required permissions," you can use Permview.exe to determine the permissions the code in you executable must receive before it will execute. Reflection emit is a runtime feature that allows code to create dynamic assemblies, modules, and types. You can dynamically create instances of these types to use, or you can use reflection emit to generate an assembly and persist it to disk as an .exe file or DLL. ReflectionPermissionAttribute Class Allows security actions for ReflectionPermission to be applied to code using declarative security. This class cannot be inherited. For a list of all members of this type, see ReflectionPermissionAttribute Members. System.Object System.Attribute System.Security.Permissions.SecurityAttribute System.Security.Permissions.CodeAccessSecurityAttribute System.Security.Permissions.ReflectionPermissionAttribute <AttributeUsage(AttributeTargets.Assembly Or AttributeTargets.Class _ Or AttributeTargets.Struct Or AttributeTargets.Constructor Or _ AttributeTargets.Method)> <Serializable> NotInheritable Public Class ReflectionPermissionAttribute Inherits CodeAccessSecurityAttribute Remarks The scope of the declaration that is allowed depends on the SecurityAction that is used. The security information declared by a security attribute is stored in the metadata of the attribute target and is accessed by the system at run time. Security attributes are used only for declarative security. For imperative security, use the corresponding permission class.

Example
The following example of a declarative attribute shows the correct way to request ReflectionPermission for ReflectionEmit and states that you must have at least this permission to run your code. <Assembly: ReflectionPermissionAttribute(SecurityAction.RequestMinimum, _ ReflectionEmit := True)>'In Visual Basic, you must specify that you are using the assembly scope when making a request. **ReflectionPermission** at link time. Demands are typically made in managed libraries (DLLs) tohelp protect methods or classes from potentially harmful code. <ReflectionPermissionAttribute(SecurityAction.Demand, _ Unrestricted := True)> Public Class SampleClass Since you do not necessarily have control over what permissions are assigned to the code you write, the common language runtime provides a mechanism for requesting the permissions that you feel your code must have in order to run properly. If the code is not granted the required permissions, it will not run. And, because permission requests are stored in an assembly's manifest, the end user can run a tool to determine what permissions have been requested by the assembly author and then take the appropriate steps

to grant those permissions if they need the code to run on their machine.

Three types of permission requests are supported: **RequestMinimum**: The permissions the code must have to run properly. If these permissions cannot be granted, the code will not be executed. **RequestOptional**: The permissions that should be granted if allowed by policy. The runtime will attempt to execute code even if permissions it requests as optional have not been granted.

RequestRefuse: The permissions that code should never be granted. Code will not receive these permissions, even if they would normally be granted to it. This is an extra precaution you can take to prevent your from code being misused. Permission requests can only be made in a declarative fashion and must always be at the assembly level (the assembly is the unit to which permissions are granted by the security system). The following code is a request stating that an assembly must have unrestricted access to the file system in order to function. [assembly:FileIOPermission(SecurityAction.RequestMinimum, Unrestricted=true)] <Assembly: FileIOPermission(SecurityAction.RequestMinimum, Unrestricted := True)> Public Class FileMover 'something interesting End Class Several requests of the same type can be made, in which case the final permission set requested is the aggregate of all requests of that type. In the example below RequestMinimum is used twice with different permissions to state that the assembly must have the ability to use Reflection Emit and perform serialization in order for it to function.

```
[assembly:ReflectionPermission(SecurityAction.RequestMinimum, ReflectionEmit=true)]
```

```
[assembly:SecurityPermission(SecurityAction.RequestMinimum, SerializationFormatter=true)] <Assembly:
```

```
ReflectionPermission(SecurityAction.RequestMinimum, ReflectionEmit := True)> <Assembly:
```

```
SecurityPermission(SecurityAction.RequestMinimum, SerializationFormatter := True)> Public Class CodeGenerator 'something interesting End Class
```

The same permission can also appear in requests of different types. For instance, the example program at the bottom of this page uses an EnvironmentPermission in each of its three requests (Minimum, Optional, an Refuse). This is useful when a permission encompasses a number of operations and you want to ensure that your assembly has the ability to perform some of those operations while being prevented from performing others. It is important to note that any permission you refuse using RequestRefuse will not be granted to your assembly even if you request that same permission using RequestMinimum. In addition to requesting individual permissions, entire sets of permissions can be requested in a compact fashion. The example below shows two requests: one stating that an assembly must have unrestricted access to the file system in order to function and one stating that it will take any and all other permissions that the security system is willing to grant it. [assembly:FileIOPermission(SecurityAction.RequestMinimum, Unrestricted=true)] [assembly:PermissionSet(SecurityAction.RequestOptional, Name="FullTrust")] <Assembly: FileIOPermission(SecurityAction.RequestMinimum, Unrestricted := True)> <Assembly: PermissionSet(SecurityAction.RequestOptional, Name := "FullTrust")> Public Class FileMover 'something interesting End Class The previous example shows how to request a permission set by name, but it is also possible to use a custom permission set representing the exact permissions you want.

Question: 14

You are an application developer for Company.com. Users who are temporary employees are members of a group named TemporaryEmployees. You develop a serviced component named CompanyComponent. CompanyComponent is part of a COM+ application named MyApplication. CompanyComponent is secured by using the SecurityRole attribute for the Employees role. You need to ensure that members of the TemporaryEmployees group are assigned to the Employees role. You decide to add the TemporaryEmployees group to the existing Employees role. Which tool should you use?

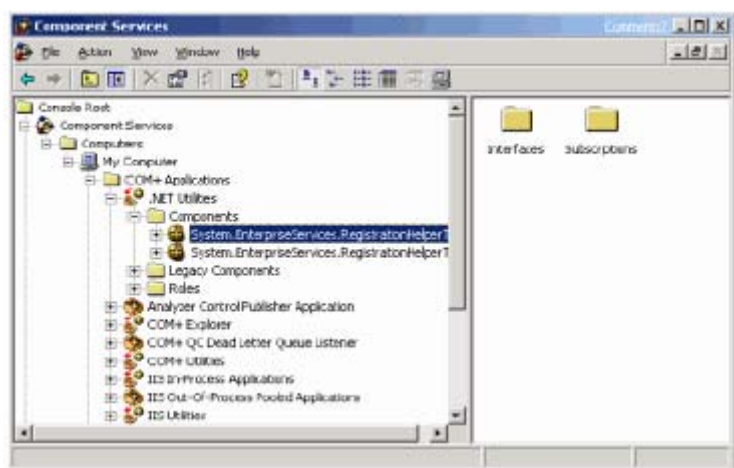
- A. The code Access Security Policy tool.
- B. The Permission View tool.
- C. The Component Services tool.
- D. The Secutil tool.
- E. The Microsoft .NET Framework Configuration tool.

Answer: C

Explanation

The Component Services tool is an administrative console for managing components registered to take advantage of component services. This tool can be found under Administrative Tools, either from the Programs Menu or the Control Panel. Components are grouped into COM+ Applications. New applications can be created by right-clicking the COM+ Applications folder and selecting New-

>Application. Components can be added to an Application by either dragging and dropping the dll, or by right-clicking the Components folder under the application and selecting New->Component. This will launch the Component Install Wizard. The following screenshot shows the Component Services management console from a Windows .NET Server machine.



Defining Roles for an Application

You determine a security policy for an application by defining the security privileges that it requires. To do this you declare a symbolic level of privilege as a role—that is, define the role for the application—and then assign the role to specific resources within the application. This design is fulfilled when the application is deployed and system administrators populate the role with actual users and user groups. For greater detail, see Role- Based Security.

To add a role to an application

- 1 In the console tree of the Component Services administrative tool, locate the COM+ application to which you want to add the role. Expand the tree to view the folders for the application.
- 2 Right-click the Roles folder for the application, point to New, and then click Role.
- 3 In the Role dialog box, type the name of the new role in the box provided.
- 4 Click OK. Note After adding roles to the application, you must be sure to assign the roles to the appropriate components, interfaces, and methods. Otherwise, if role-based security has been chosen and enabled and if roles have been added but not assigned, all calls to the application will fail. Assigning Roles to Components, Interfaces, or Methods You can explicitly assign a role to any item within a COM+ application that is visible through the Component Services administrative tool. Doing so ensures that any users that are members of the role will be permitted access to that item and any other items that it contains. For example, if you assign the role "Readers" to a component, any member of "Readers" is allowed access to that component and any interfaces and methods it exposes. "Readers" will show up as an Inherited role for any of those interfaces and methods. A method is accessible to callers only if you assign a role to it, either by explicitly assigning the role directly to the method or by assigning a role to the method's interface or the method's component, in which case the role will be inherited by the method. If no role is assigned and if access checks are enabled, all calls to the method will fail.

Before you can assign a role, you must define it for the application. All roles defined for the application will appear in the Roles explicitly set for selected item(s) window on the Security tab for any components, methods, and interfaces within the application. To assign roles to a component, method, or interface

- 1 In the console tree of the Component Services administrative tool, locate the COM+ application for which the role has been defined. Expand the tree to view the application's components, interfaces, or methods, depending on what you are assigning the role to.
- 2 Right-click the item to which you want to assign the role, and then click Properties.
- 3 In the properties dialog box, click the Security tab.
- 4 In the Roles explicitly set for selected item(s) box, select the roles that you want to assign to the item.

5 Click OK. Any roles that you have explicitly set for an item will be inherited by any lower-level items it contains and will show up in the Roles inherited by selected item(s) window for those items.

Question: 15

You are an application developer for Company.com. You create a Web Forms application to track employee expense report information. Information is entered by each user and stored in a Microsoft SQL Server database. The application uses Integrated Windows authentication with impersonation enabled to communicate with the database. All users are assigned to the DataReader role and the DataWriter role in SQL Server. The employee expense report form contains client-side validation scripts and additional server controls. This form is ViewState enabled. All employee expense reports must be approved by the accounting department by using a separate form in the application before payment is made. You need to unit exam the security of the application. What should you do?

- A. Copy the ViewState information to a text file and attempt to decrypt it.
- B. Exam the application from the hosting computer and from the client computers.
- C. Create your own page that mimics the approved page and submit that page to the server.
- D. Sign on as a user in the accounting department and verify that you can approve expense reports.

Answer: C Question: 16

You are an application developer for Company.com. You are developing an application that will be used by members of three domain user groups in your company. The user groups are named Company Sales, Company Marketing, and Account Management. Each of the three user groups will have different permission within the application. You log on to your development computer by using a domain user account that is a member of only the Domain Users and the Developers domain user groups. On your development computer, you user account is a member of only the local Users group. When you finish developing the application, you need to ensure that the application runs correctly before you send the application to the company's internal software examsing department. How should you exam the application?

- A. Select one user from each of the three user groups that will run the application. Deploy the application to the client computer of each of these three users. Exam the application on each of the computers.
- B. Deploy the application to a client computer. Ask a domain administrator to place the computer's domain account into all three of the user groups that will run the application. Exam the application on the client computer.
- C. Ask a domain administrator to create a domain user account for examsing. Place the account in each of the three user groups that will run the application. Exam the application, logging on to your computer by using the exam domain user account.
- D. Ask a domain administrator to create three domain user accounts for examsing. Place one account in each of the three user groups that will run the application. Exam the application three times, logging on to your computer by using a different exam domain User account for each exam.

Answer: D

Explanation

Developers are not normally part of the user groups that will be using custom application they design. So their accounts would not be part of the domain user group(s) with access. Applications that have not been fully examsed and approved for release need to be examsed in a nonproduction environment. There is no need to exams from three different computers, users groups/user accounts have access to the application not computer accounts. Since the groups are domain group, which have different permissions, you cannot use one account to exams all three. All three groups' permissions must be examsed and each group will hold specific users. The permission rule of AGGDLP (Accounts into Global Groups into Domain Local Groups which get permissions) must be adhered to. Since we need to exams the permission and application functionality for the three groups, we need three accounts, one for each group.

Question: 17

You are an application developer for Company.com. You are developing an application that will be used both by company users and by contractors. Contractors will install the application on their own portable computers. A written company policy prohibits contractors from easily accessing or reviewing the source code of company applications. The file servers that contain the source code for the

application are configured so that only company software developers have access. You need to ensure that the contractors cannot easily access the application source code. What should you do?

- A. Run Dotfuscator Community Edition on each of the application assemblies.
- B. Apply a strong name to each of the application assemblies.
- C. Run the Code Access Security Policy tool for each of the application assemblies before distributing the application.
- D. Use Encrypting File System (EFS) to encrypt the compiled application assemblies.

Answer: A

Explanation

Dotfuscator provides all “normal” obfuscation methodologies in addition to many unique ones. No obfuscation technology is 100 percent secure. As with other obfuscators, Dotfuscator merely makes life more difficult for decompilers and disassemblers; it does not claim nor deliver 100 percent protection. Dotfuscator will rename all possible methods. Dotfuscator can rename all public, private, etc. methods that do not override a method from a non-included class. Renaming is done in such a way as to minimize the string heap, which helps to reduce executable size. PreEmptive's Dotfuscator™ for the Microsoft .NET platform helps protect your program against reverse engineering while making it smaller and more efficient. Dotfuscator has a GUI and command line interface. Dotfuscator Community Edition is accessible directly from the tools menu of Visual Studio.NET 2003.

Question: 18

You are an application developer for Company.com. Each client computer in Company runs either Microsoft Windows XP Professional or Windows 98. You are developing an application that will be used by all users in Company. Users log on to their client computers by using a domain user account that is a member of the local Power Users group and the user's computer. You log on to your Windows XP Professional computer by using a domain user account that is a member of the local Administrators group and Power Users group on your computer. When examining your application, you need to ensure that your examss accurately reflect the production environment in which the application will run. How should you exam the application?

- A. Ask a domain administrator to temporarily remove your domain user account from the local Administrators group on your computer while you are examining the application.
- B. Exam the application on your computer. When examining, log on to the computer by using a domain user account that is a member of only the local Power Users group on your computer.
- C. Deploy the application to a Windows XP Professional computer and a Windows 98 computer. Log on to each computer by using a domain user account that is a member of only the local Power Users group.
- D. Compile the assemblies of the application from the command line by running the **runas** command and specifying a domain user account that is a member of only the local Power Users group on your computer.

Answer: C

Explanation

To appropriately examine your application, it needs to be examined in an environment that mimics the production environment. In this scenario only Windows XP has a local Power Users group, but Windows 98 systems must be examined since they are deployed in the environment.

Question: 19

You are an application developer for Company.com. You develop an ASP.NET Web application that writes to an event log named EventLog1. All managers in Company will run this application. During an exam on a manager's client computer, the application fails in the following code segment. (Line numbers are included for reference only.)

```
1 Dim EventLog1 As New EventLog
2 If Not EventLog.SourceExists("CompanyWebApp") Then
3 EventLog.CreateEventSource("CompanyWebApp", "Application")
4 End If
5 EventLog1.Source = "CompanyWebApp"
```

6 EventLog1.WriteEntry("The event occurred.")

You need to ensure that event data is written to EventLog1. You want to achieve this goal without granting unnecessary permissions.

What should you do?

- A. Insert the following code into the application. Dim eventLogDir As String eventLogDir =
 "C:\%windir%\system32\config\AppEvent.Evt" Dim FilePermission As _ New
 FileOPermission(FileIOPermissionAccess.AllAcces, eventLogDir) FilePermission.Assert()
- B. Replace line 6 of the code segment with the following line of code. EventLog1.WriteEntry("The event
 occurred","EventLogWriter")
- C. Grant the managers the Full Control permission for the event log file.
- D. Add the aspnet_wp account to the Administrators group.
- E. Create the event log source in the installer class of the application.

Answer: E

Explanation Writing Entries to Event Logs

When you write an entry to an event log, you specify the message you want to write to the log as a string. A message should contain all information needed to interpret what caused the problem and what to do to correct the problem. There are two ways you can write an entry to the log; both are equally valid. The most direct way is to register an event source with the log to which you want to write, then instantiate a component and set its Source property to that log, and finally call WriteEntry. If you do this, you do not have to set the Log property for the component instance; the log is automatically determined when you connect to a source that has already been registered. For more information on registering a source, see Adding Your Application as a Source of Event Log Entries. The other way to approach this process is to instantiate an EventLog component, set its Source, Machine, and Log properties, and call the WriteEntry method. In this case, the WriteEntry method would determine whether the source already existed and register it on the fly if it did not. The following conditions must be met in order to successfully write a log entry:

The source must be registered with the desired log. Note You must set the Source property on your EventLog component instance before you can write entries to a log. When your component writes an entry, the system automatically checks to see if the source you specified is registered with the event log to which the component is writing, and calls CreateEventSource if needed.

The message you specify cannot be more than 16K in length.

Your application must have write access to the log to which it is attempting to write. For more information, see Security Ramifications of Event Logs. You can specify several parameters when you write an entry, including the type of entry you're making, an ID that identifies the event, a category, and any binary data you want to append to the entry. For more information on the properties associated with an entry, see EventLog Members.

To write an event log entry

- 1 Instantiate an EventLog component. For more information, see Creating EventLog Component Instances.
- 2 Use the EventLog.CreateEventSource method to register an event source with the log to which you want to write an entry, using a unique string for the source string. Set the Source property for the component to the source you registered. For more information, see Configuring EventLog Component Instances. Call the WriteEntry method to specify the entry to be written to the log.
- 3 If Not EventLog.SourceExists("MyApp1") Then
- 4 EventLog.CreateEventSource("MyApp1", "Application")
- 5 End If
- 6 EventLog1.Source = "MyApp1"
- 7 EventLog1.WriteEntry("This is a simple event log entry")

Question: 20

You are an application developer for Company.com. You are developing an application that stores and retrieves data in a Microsoft SQL Server database. The application accepts input from the user and stores the input in a variable named strInput. The input is to be saved in a SQL Server nchar column that has a length of 10. The application calls a stored procedure to save the input, and the

stored procedure stores the input in a parameter named @CompanyInput. You need to ensure that input by the user can be stored in the SQL Server column without causing an exception. Your solution must ensure that either the entire user input is saved, or none of the user input is saved. Which two actions should you perform? (Each correct answer presents part of the solution. Choose two)

- A. In the application, reject the input if the strInput.Length value is greater than 10.
- B. In the application, pass only strInput.Chars(9) to the stored procedure.
- C. In the application, use a regular expression to remove all non-alphanumeric characters.
- D. In the stored procedure, reject the input if the length of @CompanyInput is greater than 10.
- E. In the stored procedure, save only the result of LEFT("CompanyInput,10) to the database column.
- F. In the stored procedure, save the result of REPLACE(@CompanyInput,"@-]",",") to the database column.

Answer: A, C

Explanation

It is best for the approach that you must validate and cleanse data before it is used and/or store. Rule number two is: *data must be validated as it crosses the boundary between untrusted and trusted environments*. By definition, trusted data is data you or an entity you explicitly trust has complete control over; untrusted data refers to everything else. In short, any data submitted by a user is initially untrusted data. When it comes to SQL statements, all dynamic SQL is bad, and parameterized store procedures must be used. Dynamic SQL can be easily compromised and used for SQL injection attacks. Validate SQL Input and Use Parameter Objects. Validate all the input data that you use in SQL commands. Do not permit the client to retrieve more data than it should. Also, do not trust user input, and do not permit the client to perform operations that it should not perform. Doing so helps to lower the risk of SQL injection. By rejecting invalid data early before you issue a command that has the invalid data, you can improve performance by eliminating unnecessary database requests. Use Parameter objects when you build database commands. When you use Parameter objects, each parameter is automatically type checked. Checking the type is another effective countermeasure you can use to help prevent SQL injection. Ideally, use Parameter objects in conjunction with stored procedures to improve performance. For more information about using parameters, see "Parameters" later in this chapter.

Using Parameters with Stored Procedures

The following code sample illustrates how to use the Parameters collection. `SqlDataAdapter myCommand = new SqlDataAdapter("AuthorLogin", conn); myCommand.SelectCommand.CommandType = CommandType.StoredProcedure; SqlParameter parm = myCommand.SelectCommand.Parameters.Add( "@au_id", SqlDbType.VarChar, 11); parm.Value = Login.Text;` In the code sample, the @au_id parameter is treated as a literal value and not as code that can be run. Also, the parameter is checked for type and length. In the code fragment, the input value cannot be longer than 11 characters. If the data does not conform to the type or length that is defined by the parameter, an exception is generated. Using stored procedures alone does not necessarily prevent SQL injection. The important thing to do is use parameters with stored procedures. If you do not use parameters, your stored procedures may be susceptible to SQL injection if the stored procedures use unfiltered input. For example, the following code fragment is susceptible to SQL injection. `SqlDataAdapter myCommand = new SqlDataAdapter("LoginStoredProcedure " + Login.Text + "", conn);`

Using Parameters with Dynamic SQL

If you cannot use stored procedures, you can still use parameters with dynamic SQL as shown in the following code fragment. `SqlDataAdapter myCommand = new SqlDataAdapter( "SELECT au_lname, au_fname FROM Authors WHERE au_id = @au_id", conn); SqlParameter parm = myCommand.SelectCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11); parm.Value = Login.Text;`

A. Retrieve Only the Columns and Rows You Need Reduce unnecessary processing and network traffic by retrieving only the columns and the rows you need. Do not use the SELECT * query. This is poor practice because you might not know the schema, or it might change. It is easy to retrieve more data than you expect. Consider a scenario where you want four columns, but you perform an operation by using the SELECT * query on a 400-column table. In that scenario, you receive many more results than you expect. Instead, use WHERE clauses to filter the rows. NCHAR - Fixed-length Unicode data with a maximum length of 4,000 characters. Returns the Unicode character with the given integer code, as defined by the Unicode standard.

Syntax

NCHAR (integer_expression)

Arguments

integer_expression Is a positive whole number from 0 through 65535. If a value outside this range is specified, NULL is returned.

Return Types

nchar(1)

Question: 21

You are an application developer for Company.com. You develop a Windows Forms application that connects to a local Microsoft SQL Server database by using the Microsoft .NET Framework Data Provider for SQL Server. The application currently connects to the database by using an account that is a member of the System Administrator role in SQL Server. You need to ensure that the application can connect to the database by using the user account of the interactive user without providing additional permissions. What should you do?

- A. Modify the application to activate a SQL Server application role.
- B. Modify the application to use SQL Server integrated security.
- C. Modify the application to send a security token that contains the authentication information in a Kerberos ticket.
- D. Modify the application to use a COM+ security roles.

Answer: B

Explanation

SQL Server provides three different security modes for validating logon information.

Security Mode	Description
Standard Security	The SQL Server user ID and password must be provided by the application. No attempt is made to use Windows NT client identification. SQL Servers prior to version 4.2 use this option exclusively.
Windows NT Integrated Security	The SQL Server user ID is always taken from the Windows NT domain user ID and password.
Mixed Security	Unless the client provides a user ID and password, the SQL Server user ID is taken from the Windows NT domain user ID and password.

Standard Security Standard security is the default installation option for SQL Server. It provides the simplest security model because security exists independently of the Windows NT domain model. You control the level of access a user has to the database and its objects by setting security options within SQL Server itself. Standard security means the SQL Server uses its own validation process for checking all logon accounts. With standard security, each user provides an additional valid SQL Server logon ID and password. Each SQL Server logon specifies the allowed access to each database and its objects (tables, views, stored procedures, and rules). The logon accounts are considered valid if they appear in the encrypted *syslogins* table. Authentication consists of comparing the provided user name and password against similar information maintained in the SQL Server database. This is the easiest security model to integrate with IIS. You must use standard security if you are not using multi-protocol or Named Pipes. Standard security works for all network configurations. With standard security, SQL Server does not consider the domain the network logon accounts are using, and also ignores the Windows NT user name and password scheme. Standard security is the best choice if your application uses Internet Information Server. An important part of this configuration is whether or not an authenticated protocol will be used. **Windows NT Integrated Security** With integrated security, SQL Server leverages Windows NT authentication to validate SQL Server logon accounts. This allows the user to bypass the standard SQL Server logon process. With this approach, a network user can access a SQL Server

database without supplying a separate logon identification or password because SQL Server obtains the user and password information from the Windows NT network security process. Integrated security works for all trusted connections and requires either Named Pipes or Multi-Protocol (with Named Pipes). Trusted connections can be from other Windows NT, Windows 95, or Windows for Workgroups workstations, and also from Microsoft LAN Manager running under either MS-DOS or Microsoft Windows clients. SQL Server applications with integrated security benefit from all of the Windows NT security features. This includes domain-wide user accounts, encrypted passwords, password aging, logon auditing, and general user administration. Integrated security requires working closely with the network administrator to grant the necessary access permissions according to your application's security model. **Note** If your application uses Internet Information Server, you probably will not use SQL Server integrated security. You can implement integrated security with SQL Server by creating several Windows NT security groups and granting each group the necessary data access rights. For example, you would grant the SystemUsers group permission to perform SELECT, UPDATE, INSERT, and DELETE using normal application processes. Similarly, you would grant the SystemAdmin group full SQL Server administrator rights and permissions. **Note** Regardless of the SQL Server's logon security mode, ODBC and DB-Library client applications can be configured to always request a trusted connection from the server. The benefit is that with a correctly configured Windows NT account the SQL Executive can connect to remote servers. Mixed Security The mixed security mode allows validation by using either standard or integrated security modes. With mixed security mode, SQL Server uses integrated security for all trusted connections. For example, for a connection to a trusted ODBC source, authentication occurs via the Windows NT authentication process. If the integrated mode authentication fails, standard security mode is used requiring the entry of valid SQL Server logon information

Database Security

One of the most common scenarios for a distributed application involves reading and writing data on a remote database. The dilemma that arises is how to do so securely while maintaining application scalability. Where you choose to manage security in your application will greatly impact, either negatively or positively, the scalability of your application. To achieve scalability using database connection pooling foregoes having the database manage security. This is because database connection pooling requires the connection string be identical to pool connections. Therefore, you must manage security elsewhere. If you must track database operations on per user basis, consider adding a parameter for user identity to each operation and manually log user actions in the database. Following the advice above, another issue is how to store the database connection string, which typically contains security credentials, so multiple users can access it without compromising security. Most sample applications demonstrate storing the connection string in the Web.config or global.asax files. However, because these files are plain text files that have limited security, it is not the best location for storing this information. Should an intruder compromise your Web server's security, these files would be easily accessible. Here are just a few alternatives:

- If using the Web.config file, store the connection string encrypted and then decrypt the connection string in your application code when needed.

- Build a COM+ application using the ServicedComponent Class and store the connection string in the construct string for that component. When storing sensitive information in the constructor string, you should verify the following:

- Only the appropriate users/groups belong to the Reader role of the System Package. However, you must carefully manage COM+ to prevent it from being unable to read its own configuration.

- You have controlled and audited access to the %windir%\Registration folder, where the COM+ configuration database (RegDB) stores its files. For more information, see ServicedComponent Class.

- Use integrated security to make a trusted connection with SQL Server. This makes it possible for you to use a connection string that eliminates the need for storing a password in the connection string, such as: "Data Source=mySqlServer;Integrated Security=SSPI;Initial Catalog=myDB" There are some drawbacks to using integrated security, most of which you can overcome. Because integrated security requires a Windows account, it defeats connection pooling if you impersonate each authenticated principal using an individual Windows account. However, if you instead impersonate a limited number of Windows accounts, with each account representing a particular role, you can overcome this drawback. Each Windows account must be a domain account with IIS and SQL Server in the same or trusted domains. Alternatively, you can create identical (including passwords) Windows accounts on each machine. After a typical installation, the default security authentication mode is Windows Authentication for SQL

Server 2000, which is different from SQL Server 7.0. In SQL Server 7.0, the default authentication mode is Mixed (Windows Authentication Mode and SQL Server Authentication). Windows Authentication is a better security method because of the additional security features it provides, such as secure validation and encryption of passwords, password expiration and auditing. For more information, see Authentication Modes. If you configure SQL Server to use Windows Authentication, you could create one Windows account for read-only operations and another Windows account for read/write operations. You then map each Windows account to a SQL Server login and establish the desired permissions. Using application logic, you then determine which Windows account to impersonate when performing database operations. In SQL Server, you can add any Windows user account as a member of a fixed database role. Each member gains the permissions applied

to the fixed database role. For more information, see Managing Permissions. For SQL Server 7.0, integrated security does not work with SQL Server's TCP/IP network library, but uses the named pipes network library instead.

As an added security measure, the `ConnectionString` property of the `SqlConnection` object does not persist or return the full connection string by default. To do so, you must set `Persist SecurityInfo` to `true`

Question: 22

You are an application developer for your company. You are conducting a code review of a Windows Forms application that was developed by another developer. The application includes a function named `Logon()`, which validates a user's logon credentials. The function displays a dialog box for the user to enter the user's credentials, and the function validates those credentials by using a database. The function returns a value of 0 if the user's password is incorrect, a value of 1 if the user's user ID is incorrect, and a value of 2 if both are correct. Users should receive access to the application only if the function returns a value of 2. A function named `EndApp()` is used to exit the application. The application must display a message to the user, depending on the result of the `Logon()` function. The application contains the following code segment.

```
Dim intResult As Integer = Logon()
Select Case intResult
Case 0
    MsgBox "User name is OK, password incorrect."
Case 1
    MsgBox "User name is incorrect."
Case Else
    MsgBox "Welcome!"
End Select
If intResult <> 2 Then EndApp()
End If
```

You need to improve the security of this code segment while maintaining its functionality. You decide to replace the existing code segment. Which code segment should you use?

- A.

```
If Logon() <> 2 Then Console.WriteLine "Logon error." EndApp()
End If
```
- B.

```
If Logon() <> 2 Then Console.WriteLine "Logon error." EndApp()
Else MsgBox "Welcome!"
End If
```
- C.

```
Dim intResult As Integer = Logon()
Select Case intResult
Case 0
    MsgBox "User name is OK, password incorrect." EndApp()
Case 1
    MsgBox "User name is incorrect." EndApp()
Case Else
    MsgBox "Welcome!"
End Select
```
- D.

```
Dim intResult As Integer = Logon()
If intResult = 2 Then MsgBox "Welcome!"
Else MsgBox "User name or password was incorrect." EndApp()
End If
```

Answer: D

Question: 23

You are an application developer for your company. You are conducting a code review of an application that was developed by another developer. The code declares a variable named `intPermissions` and a variable named `strGroups`. A portion of the application code defines security permissions for the user. The application is designed so that `intPermissions` contains an integer that indicates various permissions within the application, and `strGroups` contains the name of a user group. The `intPermissions` variable also contains values that indicate other information about the user. The `strGroups` and `intPermissions` variables are initially populated by other components, which are called by the main application. The application contains the following code segment. (Line numbers are included for reference only.)

```
1 Select Case strGroups
2 Case "Administrator", "Admin"
3     intPermissions = intPermissions OR 256
4 Case "Reviewer"
5     intPermissions = intPermissions OR 128
6 Case "Manager"
7     intPermissions = intPermissions OR 64
8 End Select
```

The design document for the application states that `intPermissions` must have a value of zero when the user has no permissions. The design document also states that users not belonging to one of the four predefined groups must have no permissions. You need to ensure that the code segment assigns the correct value to `intPermissions` in all circumstances. What should you do?

- A. Add the following code before line 1 of the code segment.
- ```
If intPermissions = 0 Then Throw New ApplicationException
End If
```

- B. Add the following code before line 1 of the code segment. `intPermissions = 0`
- C. Add the following code between lines 1 and 2 of the code segment. `Case "" intPermissions = 0`
- D. Add the following code between lines 7 and 8 of the code segment. `Case Else intPermissions = 0`

**Answer: B**

#### **Question: 24**

You are an application developer for Company.com. You are developing the business layer of a three-tier Web application. The application uses Forms authentication. File access permissions are assigned based on the role of the user. The authenticated user names and roles are stored in a Microsoft SQL Server database. When a user is authenticated by the server, the user's cryptographically random session token and role are stored in a cookie on the client computer and used for access to other pages. You want users who are members of a role name Company Staff to have Read permission and Write permission for the application files. You want users who are members of a role named Reader to have only Read permission for the files. You create a method named `OpenFile` to pass the name and role of the current user along with the name of a file. This information is used to open and return a file. The method is contained in the following code segment.

```
Public Function OpenFile(ByVal UserInfo As HttpCookieCollection, _
 ByVal File As String) As Stream Select Case
 UserInfo("Role").Value Case "CompanyStaff" Return New FileStream(File, FileMode.Open, FileAccess.ReadWrite) Case Else Return
 New FileStream(File, FileMode.Open, FileAccess.Read) End Select End Function
```

During a security review, you discover that some users will receive Write permission when they should not. You need to prevent unauthorized users from modifying files. What should you do?

- A. Change the application to use Windows authentication.
- B. Create a restrictive access control list (DACL) entry for each file.
- C. Verify the role by using the word Reader, instead of relying on the default case.
- D. Retrieve the user's role information from the database instead of from the cookie.

**Answer: D**

#### **Explanation**

Cookies are a prime target for session high-jacking. There should be limited information stored in them as they can be manipulated client-side if stored there. Since the role is stored here in plain text it can just be changed to a higher role. The question only ask for one option but it specifically states that one role has read/write and the other only has read, but the code only checks for the read/write role, which would mean everyone using the code would at least have read when in only the one role has read only. It would be best to not only pull the role from a more secure/securable source other than the cookie but also check for the roles in the case select. Bulletproof persistent cookies to increase security Web browser cookies can enhance the user experience by providing additional functionality and ease of use. However, from an administration point of view, cookies are a security concern. Encrypt your cookies with this simple technique.

Cookies offer an excellent way to keep small bits of information about the user readily available so that they don't have to be looked up again. They can allow you to keep users' numeric IDs handy instead of their logon names, which makes getting back to their security and authorization easier and quicker. However, cookies represent a dangerous risk: Users may choose to tamper with the information and see what havoc it might cause.

#### **Risks of cookie tampering**

To understand how using cookies can create huge security risks, consider a site that stores the user's ID in a database in a cookie. The cookie is persistent, and the site never validates the information in the cookie. It's assumed to be correct. The user goes to the site and logs in. He or she looks at the cookie file and determines that only an ID is being stored in it. The user resets the number in the file to 1 and logs back into the site. For most sites, the super user is the first ID inserted into the database, and it's usually never disabled. If the site doesn't validate the value from the cookie, the user has become a complete administrator with a trivial amount of text editing. Any unchecked information placed in a cookie can represent a potential security problem.

**Question: 25**

You are an application developer for Company.com. To prevent malicious code from running, a written company policy does not permit developers to log on by using accounts that have more permissions than necessary. Your user account is a member of the Users group and the VS Developers group. You attempt to run an application that requires Administrator-level permissions. You receive an error message that states that permission is denied. You need to be able to run the application. What should you do?

- A. Ask the network administrator to add your user account to the domain Administrators group.
- B. Ask the administrator of your client computer to add your user account to the local Administrators group.
- C. Add the administrator of your client computer to add your user account to the Power Users group.
- D. Run the application by using the **runas** command and specify a user account in the local Administrators group.

**Answer: D Explanation Run Using a Least-Privileged Account**

You should develop applications using a non administrator account. Doing so is important primarily to limit the exposure of the logged on user and to help you to design more secure software. For example, if you design, develop, and exams an application while you are interactively logged in as an administrator, you are much more likely to end up with software that requires administrative privileges to run. You should not generally log on using the local administrator account. The account that you use on a daily basis should not be a member of the local Administrators group. Sometimes you might still need an account that has administrative privileges—for example, when you install software or edit the registry. Because the default local administrator account is well known, however, and it is the target of many attacks, create a nonstandard administrator account and use this only when it is required. To create accounts for development Remove your current user account from the Administrators group if it is a member. Create a new custom administration account using a nonstandard name and strong password. Use your non-administrator account to logon interactively on a daily basis. When you need to run a command with administrative privileges, use your custom administration account with the Runas.exe command line utility.

**Running Privileged Commands**

To run a privileged command, you can use one of the following techniques to temporarily change your security context: Use the Runas.exe utility from a command line. The following command shows you how to use the Runas.exe utility to launch a command console that runs under your custom administration account. `runas.exe /user:mymachine\mycustomadmin cmd.exe` By executing `Cmd.exe`, you start a new command window that runs under the security context of the user you specify with the `/user` switch. Any program you launch from this command window also runs under this context. Use Run As from Windows Explorer. You can right-click an executable file in Windows Explorer and click Run As. To display this item on Windows 2000, hold the shift key down and then right-click an executable file. When you click Run As, you are prompted for the credentials of the account you want to use to run the executable file. Use Run As shortcuts. You can create quick launch and desktop shortcuts to easily run applications using a privileged user account. The following example shows a shortcut that you can use to run Windows Explorer (`Explorer.exe`) using the administrator account: `%windir%\System32\runas.exe /user:administrator explorer` Note If using a non-administrator account proves impractical for your environment, still exams your application or component while running as a least privileged user to catch and correct problems before deploying. For example, your application might incorrectly require administrator privileges without your realizing it, which would cause the application to fail when it is deployed in a production environment.

**Question: 26**

You are an application developer for Company.com. You are using the Microsoft .NET Framework to develop an application that uses a Web service. The Web service is provided by a vendor and is accessed over the Internet.

Your application retrieves string data from the Web service and stored it in a variable named `strCompanyData`. The application also defines a `SqlCommand` object named `objCmd`. The application contains the following code segment.

```
Dim strQuery As String
strQuery = "INSERT INTO WebTable (WebData) VALUES(" & strQuery & " & strCompanyData & ")"
objCmd.CommandText = strQuery
objCmd.ExecuteNonQuery()
```

You need to improve the security of this code segment while maintaining its functionality.

What should you do?

- A. Modify the application to use declarative security.
- B. Ask the vendor to provide a Web service that is written by using the .NET Framework.
- C. Ask the vendor to perform data validation on all data that is provided by the Web service.
- D. Format the contents of strCompanyData to be compatible with the SQL Server data type and remove encoded data or SQL statements.

**Answer: D**

### Explanation

It is best for the approach that you must validate and cleanse data before it is used and/or store. Rule number two is: *data must be validated as it crosses the boundary between untrusted and trusted environments*. By definition, trusted data is data you or an entity you explicitly trust has complete control over; untrusted data refers to everything else. In short, any data submitted by a user is initially untrusted data. When it comes to SQL statements, all dynamic SQL is bad, and parameterized store procedures must be used. Dynamic SQL can be easily compromised and used for SQL injection attacks. All relational databases—including SQL Server, Oracle, IBM DB2, and MySQL—are susceptible to SQL injection attacks. You can buy products that protect your system from SQL injection, but for most businesses, the defense against SQL-injection attack must be code-based. The opening for SQL-injection attacks comes primarily through Web applications that combine user input with dynamic SQL to form SQL commands that the application sends to the database. Here are four important steps you can take to protect your Web applications from SQL-injection attacks. In addition to the following tips, the Microsoft Patterns and Practices Library that I highlighted last month provides advice about securing your data-access applications.

4. Principle of Least Privilege The account an application uses to connect to the database should have only the privileges that application requires. The security permissions that an intruder gains from a compromised application define the harm that the intruder can inflict. Applications shouldn't connect as sa or with the Administrator account. Instead, the account should have permissions to access only the database objects it needs.

3. Validate All Input If an input field should contain numeric data, then verify that users enter only numbers. If character data is acceptable, check for unexpected characters. Make sure your application looks for characters such as semicolons, equals signs, double dashes, brackets, and SQL keywords. The .NET Framework provides regular expressions that enable complex pattern matching, a good way to exams user input. Limiting the length of accepted user input is also a good idea. Validating your input might seem obvious, but many applications are vulnerable to SQL-injection attacks because intruders can use the openings that Web applications offer.

2. Avoid Dynamic SQL Dynamic SQL is a great tool for performing ad hoc queries, but combining dynamic SQL with user input creates exposure that makes SQL-injection attacks possible. Replacing dynamic SQL with prepared SQL or stored procedures is feasible in most applications. Prepared SQL and stored procedures accept user input as parameter data rather than as SQL commands, thus limiting what an intruder can do. Of course, replacing dynamic SQL with a stored procedure won't help you if you use the user input to build dynamic SQL statements in your stored procedures. In that case, the dynamic SQL that the user input creates will still be corrupted, and your database will still be in danger of SQL-injection attack.

1. Use Double Quotes Replace all the single quotes that your users' input contains with double quotes. This simple precaution will go a long way toward warding off SQL-injection attacks. Single quotes often terminate SQL expressions and give the input more power than is necessary. Replacing the single quotes with double quotes will cause many SQL injection attacks to fail.

### Question: 27

You are an application developer for Company.com. You are developing a three-tier application. You enter sample data to exam the application. The following exception is caught by the data layer before the application continues to run.

Cannot set column 'Column1' to 'Text too long for maximum length'. The value violates the MaxLength limit of this column

You need to improve the security of the application.

Which two actions should you perform? (Each correct answer presents part of the solution. Choose two)

- A. Increase the maximum length of data characters allowed in the column.
- B. Validate all incoming data character lengths at the business layer.
- C. Modify the data layer to process data above the maximum length.
- D. Modify the user interface to prevent users from entering data above the maximum character length.

**Answer: B, D**

**Question: 28**

You are an application developer for your company. You are developing an ASP.NET Web application that uses Integrated Windows authentication to identify users. When a user runs the application for the first time, a new record is created in a user database. The application must work for both local users and domain users. All domain user accounts are created in the default user container. You need to develop code that will generate the primary key to store in the user database to uniquely identify each user. Which code segment should you use?

- A. `ReadOnly Property PrimaryKey() As String Get Dim Wi As WindowsIdentity = WindowsIdentity.GetCurrent() Dim Username As String = Wi.Name.ToUpper() Dim Sep As Integer = Username.IndexOf("\") If Sep > 0 Then Return Username.Substring(Sep + 1) End If Return Username End GetEnd Property`
- B. `ReadOnly Property PrimaryKey() As String Get Dim Wi As WindowsIdentity = WindowsIdentity.GetCurrent() Return Wi.Name.ToUpper() End GetEnd Property`
- C. `ReadOnly Property PrimaryKey() As String Get Dim Wi As WindowsIdentity = WindowsIdentity.GetCurrent() Dim Username As String = Wi.Name.ToUpper() Dim Sep As Integer = Username.IndexOf("\") If Sep > 0 Then Return Username.Substring(Sep + 1) & "@" & Username.Substring(0, Sep) End If Return Username End GetEnd Property`
- D. `ReadOnly Property PrimaryKey() As String Get Dim Wi As WindowsIdentity = WindowsIdentity.GetAnonymous() Dim Username As String = Wi.Name.ToUpper() Return Username End GetEnd Property`

**Answer: B**

**Question: 29**

You are an application developer for Company.com. You develop an application that customers will be able to automate by using Microsoft Visual Basic for Applications (VBA) scripts. The application will be accompanied by sample VBA scripts.

Customers must be able to review the sample VBA scripts. You want customers to be able to automate the installed application by using any of the sample VBA scripts or by creating their own automation scripts. You also want to allow customers to choose not to apply any automation scripts.

You need to distribute the sample VBA scripts with your application in a manner that minimizes security risks for the customer.

What should you do?

- A. On installation, place all the sample VBA scripts in a subfolder of the application's installation folder.
- B. On installation, as the user to choose one sample VBA script to install as the application's automation script.
- C. Do not install the same VBA scripts. Leave the files in a folder on the installation media.
- D. Encrypt same VBA scripts on the installation media and decrypt the files during installation.

**Answer: C**

**Explanation**

Security in deployment is one of the tenets of the Microsoft Trustworthy Computing Security Framework. This section contains articles that will help you understand how to deploy your applications in a secure manner and how to use features like code access security to deploy applications in alternative security contexts. Sample applications, scripts, etc; do not go through much quality or

security review at all, if any. It is best practice not to have them auto installed with deployments as they can be used against the environment/computer. Users can always copy what they need. VBAs are very powerful and can be used against systems where they are installed, especially if they are installed without the user's full understanding of the implications of installing them. Since they are really templates, they are not required for the installation or for normal application operation in the case. Prime example is MSDE which is part of Microsoft Office 2000 and higher, it is not required and therefore is a separate install.

**Question: 30** You are an application developer for Company.com. You are developing an application. Part of the application accepts a URL from the user and stored the URL in a variable named `strInput`. Only URLs that specify HTTP or FTP as the protocol are usable by the application. URLs specifying the messenger, news, file, or other protocols are not permitted because they might allow the user to bypass certain security features. You need to ensure that the URL provided by the user specifies only HTTP or FTP as the protocol. What should you do?

- A. Exam the user's input by using the following regular expression. `^(http|ftp:)` Reject the input that does not match the regular expression.
- B. Exam the user's input by using the following regular expression. `^(messenger:|file:|news:)` Reject input that matches the regular expression.
- C. Modify the contents of `strInput` so that all instances of messenger or news are replaced with `http`, and all instances of file are replaced with `ftp`.
- D. Add the following code segment to the application. If `strInput.Chars(0) <> "h"` And `strInput.Chars(0) <> "f"` Then  
    `MsgBox "Protocol is not allowed."`  
    `strInput = ""`  
    End If
- E. Add the following code segment to the application. Select Case `StrInput` Case "messenger", "news", "file" `MsgBox "Protocol is not allowed."` `strInput = ""`

**Answer: A**

### Explanation

It is best for the approach that you must validate and cleanse data before it is used and/or store. Rule number two is: *data must be validated as it crosses the boundary between untrusted and trusted environments*. By definition, trusted data is data you or an entity you explicitly trust has complete control over; untrusted data refers to everything else. In short, any data submitted by a user is initially untrusted data. Always exams for what is acceptable, discard the rest. As with firewalls, only what is specifically needed is allowed to pass all else is blocked/dropped. Examining for what is unacceptable is prone to error because it is not possible to exams for all things that can go wrong or that should be rejected. Regular expressions are the best way of doing this. A Regular Expression Rosetta Stone Regular expressions are incredibly powerful, and their usefulness extends beyond just restricting input. They constitute a technology worth understanding for solving many complex data manipulation problems. I write many applications, mostly in Perl and C#, that use regular expressions to analyze log files for attack signatures and to analyze source code for security defects. *Regular Expressions in Managed Code* Most if not all applications written in C#, Managed C++, Microsoft Visual Basic .NET, ASP.NET, and so on have access to the .NET Framework and as such can use the *System.Text.RegularExpressions* namespace. I've already outlined its syntax earlier in this chapter. However, for completeness, following are C#, Visual Basic .NET, and Managed C++ examples of the date extraction code I showed earlier in Perl. Imports `System.Text.RegularExpressions` Dim s As String Dim r As Regex s = "We leave at 12:15pm for Mount Doom." r = New Regex(".\*(\d{2}:\d{2}[ap]m)", RegexOptions.IgnoreCase) If r.Match(s).Success Then Console.WriteLine(r.Match(s).Result("\$1")) End If

### Regular Expressions

Regular expressions are a concise and flexible notation for finding and replacing patterns of text. The regular expressions used within Visual Studio are a superset of the expressions used in Visual C++ 6.0, with a simplified syntax. You can use the following regular expressions in the Find, Replace, Find in Files or Replace in Files dialog boxes to refine and expand your search. **Note** You must select the Use check box in the Find, Replace, Find in Files, and Replace in Files dialog boxes before using any of the following expressions as part of your search criteria. The following expressions can be used to match characters or digits in your search string: The following table lists the syntax for matching by standard Unicode character properties. The two-letter abbreviation is the same as listed in the Unicode character properties database. These may be specified as part of a character set. For example, the expression

[Nd:Nl:No] matches any kind of digit.

| Expression             | Syntax | Description                                                                                                                                                     |
|------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Any character          | .      | Matches any one character except a line break.                                                                                                                  |
| Maximal - zero or more | *      | Matches zero or more occurrences of the preceding expression.                                                                                                   |
| Maximal - one or more  | +      | Matches at least one occurrence of the preceding expression.                                                                                                    |
| Minimal - zero or more | @      | Matches zero or more occurrences of the preceding expression, matching as few characters as possible.                                                           |
| Minimal - one or more  | #      | Matches one or more occurrences of the preceding expression, matching as few characters as possible.                                                            |
| Repeat n times         | ^n     | Matches n occurrences of the preceding expression. For example, [0-9]^4 matches any 4-digit sequence.                                                           |
| Set of characters      | []     | Matches any one of the characters within the []. To specify a range of characters, list the starting and ending character separated by a dash (-), as in [a-z]. |
| Character not in set   | [^...] | Matches any character not in the set of characters following the ^.                                                                                             |
| Beginning of line      | ^      | Anchors the match to the beginning of a line.                                                                                                                   |
| End of line            | \$     | Anchors the match to the end of a line.                                                                                                                         |
| Beginning of word      | <      | Matches only when a word begins at this point in the text.                                                                                                      |

|             |    |                                                                                       |
|-------------|----|---------------------------------------------------------------------------------------|
| End of word | >  | Matches only when a word ends at this point in the text.                              |
| Grouping    | () | Groups a subexpression.                                                               |
| Or          |    | Matches the expression before or after the OR symbol ( ). Mostly used within a group. |

| Expression        | Syntax | Description                                                                                                                              |
|-------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------|
| Uppercase letter  | :Lu    | Matches any one capital letter. For example, :Luhe matches "The" but not "the".                                                          |
| Lowercase letter  | :Ll    | Matches any one lower case letter. For example, :Llhe matches "the" but not "The".                                                       |
| Title case letter | :Lt    | Matches characters that combine an uppercase letter with a lowercase letter, such as Nj and Dz.                                          |
| Modifier letter   | :Lm    | Matches letters or punctuation, such as commas, cross accents, and double prime, used to indicate modifications to the preceding letter. |
| Other letter      | :Lo    | Matches other letters, such as gothic letter ahsa.                                                                                       |
| Decimal digit     | :Nd    | Matches decimal digits such as 0-9 and their full-width equivalents.                                                                     |
| Letter digit      | :NI    | Matches letter digits such as roman numerals and ideographic number zero.                                                                |
| Other digit       | :No    | Matches other digits such as old italic number one.                                                                                      |
| Open punctuation  | :Ps    | Matches opening punctuation such as open brackets and braces.                                                                            |

|                   |     |                                                                  |
|-------------------|-----|------------------------------------------------------------------|
| Close punctuation | :Pe | Matches closing punctuation such as closing brackets and braces. |
|-------------------|-----|------------------------------------------------------------------|

|                           |     |                                                                   |
|---------------------------|-----|-------------------------------------------------------------------|
| Initial quote punctuation | :Pi | Matches initial double quotation marks.                           |
| Final quote punctuation   | :Pf | Matches single quotation marks and ending double quotation marks. |
| Dash punctuation          | :Pd | Matches the dash mark.                                            |
| Connector punctuation     | :Pc | Matches the underscore or underline mark.                         |

In addition to the standard Unicode character properties, the following additional properties may be specified. These properties may be specified as part of a character set.

| Expression            | Syntax | Description                                                                                       |
|-----------------------|--------|---------------------------------------------------------------------------------------------------|
| Alpha                 | :Al    | Matches any one character. For example, :Alhe matches words such as "The", "then", and "reached". |
| Numeric               | :Nu    | Matches any one number or digit.                                                                  |
| Punctuation           | :Pu    | Matches any one punctuation mark, such as ?, @, ', and so on.                                     |
| White space           | :Wh    | Matches all types of white space, including publishing and ideographic spaces.                    |
| Bidi                  | :Bi    | Matches characters from right-to-left scripts such as Arabic and Hebrew.                          |
| Hangul                | :Ha    | Matches Korean Hangul and combining Jamos.                                                        |
| Hiragana              | :Hi    | Matches hiragana characters.                                                                      |
| Katakana              | :Ka    | Matches katakana characters.                                                                      |
| Ideographic/Han/Kanji | :Id    | Matches ideographic characters, such as Han and Kanji.                                            |

**Question: 31**

You are an application developer for the research department of Company.com. All department applications are developed by using the Microsoft .NET Framework. Application permissions are configured by using the .NET Framework. A written company policy states that applications that are not controlled by the .NET Framework are not allowed to run on computers in the research department. All users in the research department have a separate client computer to run commercial applications that are not written by using the .NET Framework.

You are conducting a code review of a new research department application that was written by another developer. The application contains the following code segment.

```
Dim objGeneric As Object objGeneric = CreateObject("Company.ExternalData") objGeneric =
CreateObject("Company.ExternalData") objGeneric.RetrieveData() SQLSave(objGeneric.Data)
```

You discover that the SQLSave() function inserts information into a Microsoft SQL Server database. You need to ensure that this code segment complies with the written policy for client computers in the research department.

What should you do?

- A. Add a project reference for Company.ExternalData and replace the first two lines of the code segment with the following code: `Dim objGeneric As Company.ExternalData`
- B. Rewrite Company.ExternalData as a managed code component or obtain an equivalent managed code component.
- C. Move the SQLSave() function to a managed code component.
- D. Validate the contents of objGeneric.Data to remove SQL statements or encoded data.

**Answer: B**

**Explanation**

Only managed code must be used on the research computers. Since the code shows the ExternalData as a called component (assuming that this is not managed code component), we need to change it to a managed code component.

**Question: 32**

You are an application developer for Company.com. You are conducting a code review of an application that was created by another developer for in Company. The application accesses data that is stored in a Microsoft SQL Server database. The application accepts input from a Web page into a variable named strInput. The application already contains a function named CompanyInput(). The CompanyInput() function ensures that the user's input contains only alphanumeric characters and that the input contains no SQL statements, spaces, or other invalid data. The input is then saved in the SQL Server database. While examining the application, you discover that CompanyInput() is properly removing the space from input such as Hello World. However, when you enter Hello%20World, the function produces Hello20World instead of HelloWorld. You need to ensure that the invalid input is removed by the CompanyInput() function.

What should you modify the CompanyInput() function to do?

- A. Use the HttpUtility.UrlDecode method and the HttpUtility.HtmlDecode method to modify the user input.
- B. Look for percent signs in the input and remove the percent sign and the two characters following the percent sign.

- C. Remove all punctuation marks and symbols from the input.
- D. Use the following regular expression to modify the input. `[^\w\.\@-]`

### Answer: A Explanation

**HttpUtility.UrlDecode Method (Byte[], Encoding)** Converts a URL-encoded byte array into a decoded string, using the specified decoding object. **Overloads Public Shared Function UrlDecode( \_ ByVal bytes() As Byte, \_ ByVal e As Encoding \_ ) As String** Return Value The decoded string. Remarks If characters such as blanks and punctuation are passed in an HTTP stream, they might be misinterpreted at the receiving end. URL encoding converts characters that are not allowed in a URL into character-entity equivalents; URL decoding reverses the encoding. For example, when embedded in a block of text to be transmitted in a URL, the characters < and > are encoded as %3c and %3d. **HttpUtility.HtmlDecode Method (String, TextWriter)** Converts a string that has been HTML-encoded into a decoded string, and sends the decoded string to a TextWriter output stream. [Visual Basic] **Overloads Public Shared Sub HtmlDecode( \_ ByVal s As String, \_ ByVal output As TextWriter \_** Remarks If characters such as blanks and punctuation are passed in an HTTP stream, they might be misinterpreted at the receiving end. HTML encoding converts characters that are not allowed in HTML into character-entity equivalents; HTML decoding reverses the encoding. For example, when embedded in a block of text, the characters < and >, are encoded as &lt; and &gt; for HTTP transmission.

### Question: 33

You are an application developer for Company.com. You develop an application that stores and retrieves files in a folder on a file server. The files are named by users. The design specifications for the application require you to prevent users from accessing files that are stored in any other folder. You need to add a function that will prevent files that have the .LNK file name extension from being stored in the folder. Which code segment should you use?

- A. `Public Function OpenFile(ByVal Name As String) As FileStream If Name.ToLower().EndsWith(".lnk") Then Throw New ApplicationException("Unable to store LNK files") End If Return New FileStream(Name, FileMode.Create) End Function`
- B. `Public Function OpenFile(ByVal Name As String) As FileStream Dim Extension As Integer = Len(Name) - 1 Dim Nchar() As Char = Name.ToCharArray() While NChar(Extension) = "." Extension = Extension - 1 End While Name = Name.Substring(0, Extension + 1) If Name.ToLower().EndsWith(".lnk") Then Throw New ApplicationException("Unable to store LNK files") End If Return New FileStream(Name, FileMode.Create) End Function`
- C. `Public Function OpenFile(ByVal Name As String) As FileStream Dim Fi As FileInfo = New FileInfo(name) If Fi.Extension.ToLower().Equals(".lnk") Then Throw New ApplicationException("Unable to store LNK files") End If Return Fi.Open(FileMode.Create) End Function`
- D. `Public Function OpenFile(ByVal Name As String) As FileStream Dim Fi As FileInfo = New FileInfo(Name) If Fi.Attributes <> FileAttributes.Normal Then Throw New ApplicationException("Unable to store LNK files") End If`

### Answer: C

### Explanation

Get just the extension of a file from the complete path string Use FileInfo. Instantiate a FileInfo object with the full path as constructor arg. Then simply call FileInfo.Extension and you will get just the extension of the file. `FileInfo finfo = new FileInfo(strFileName); Console.WriteLine(finfo.Extension);`

### FileInfo Class

Provides instance methods for the creation, copying, deletion, moving, and opening of files, and aids in the creation of FileStream objects. For a list of all members of this type, see FileInfo Members. **System.Object System.MarshalByRefObject System.IO.FileSystemInfo System.IO.FileInfo <Serializable> NotInheritable Public Class FileInfo Inherits FileSystemInfo** Remarks Use the FileInfo class for typical operations such as copying, moving, renaming, creating, opening, deleting, and appending to files. Many of the FileInfo methods return other I/O types when you create or open files. You can use these other types to further manipulate a file. For more information, see specific FileInfo members such as Open, OpenRead, OpenText, CreateText, or Create. If you are going to reuse an object several times, consider using the instance method of FileInfo instead of the corresponding static methods of the File class, because a security check will not always be necessary. By default, full read/write access to new files is granted to all users. The following table describes the enumerations that are used to customize the behavior of various FileInfo methods.

| Enumeration | Description                                                                                                                                          |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| FileAccess  | Specifies read and write access to a file.                                                                                                           |
| FileShare   | Specifies the level of access permitted for a file that is already in use.                                                                           |
| FileMode    | Specifies whether the contents of an existing file are preserved or overwritten, and whether requests to create an existing file cause an exception. |

Note In members that accept a path as an input string, that path must be well-formed or an exception is raised. For example, if a path is fully qualified but begins with a space, the path is not trimmed in methods of the class. Therefore, the path is malformed and an exception is raised. Similarly, a path or a combination of paths cannot be fully qualified twice. For example, "c:\temp c:\windows" also raises an exception in most cases. Ensure that your paths are well-formed when using methods that accept a path string. In members that accept a path, the path can refer to a file or just a directory. The specified path can also refer to a relative path or a Universal Naming Convention (UNC) path for a server and share name. For example, all the following are acceptable paths:

"c:\\MyDir\\MyFile.txt" in C#, or "c:\\MyDir\\MyFile.txt" in Visual Basic.

"c:\\MyDir" in C#, or "c:\\MyDir" in Visual Basic.

"MyDir\\MySubdir" in C#, or "MyDir\\MySubDir" in Visual Basic.

"\\MyServer\\MyShare" in C#, or "\\MyServer\\MyShare" in Visual Basic. For an example of using this class, see the Example section below. The following table lists examples of other typical or related I/O tasks.

| To do this...                 | See the example in this topic...                                           |
|-------------------------------|----------------------------------------------------------------------------|
| Create a text file.           | Writing Text to a File                                                     |
| Write to a text file.         | Writing Text to a File                                                     |
| Read from a text file.        | Reading Text from a File                                                   |
| Append text to a file.        | Opening and Appending to a Log File<br>File.AppendText FileInfo.AppendText |
| Rename or move a file.        | File.Move FileInfo.MoveTo                                                  |
| Delete a file.                | File.Delete FileInfo.Delete                                                |
| Copy a file.                  | File.Copy FileInfo.CopyTo                                                  |
| Get the size of a file.       | FileInfo.Length                                                            |
| Get the attributes of a file. | File.GetAttributes                                                         |

|                                                   |                                                  |
|---------------------------------------------------|--------------------------------------------------|
| Set the attributes of a file.                     | File.SetAttributes                               |
| Determine if a file exists.                       | File.Exists                                      |
| Read from a binary file.                          | Reading and Writing to a Newly Created Data File |
| Write to a binary file.                           | Reading and Writing to a Newly Created Data File |
| Retrieve a file extension.                        | Path.GetExtension                                |
| Retrieve the fully qualified path of a file.      | Path.GetFullPath                                 |
| Retrieve the file name and extension from a path. | Path.GetFileName                                 |
| Change the extension of a file.                   | Path.ChangeExtension                             |

.NET Compact Framework Platform Note: The .NET Compact Framework does not support getting or setting directory attributes.

**FileStream Class**

Exposes a Stream around a file, supporting both synchronous and asynchronous read and write operations. For a list of all members of this type, see FileStream Members. System.Object System.MarshalByRefObject System.IO.Stream System.IO.FileStream System.IO.IsolatedStorage.IsolatedStorageFileStream Public Class FileStream Inherits Stream Remarks Use the FileStream class to read from, write to, open, and close files on a file system, as well as to manipulate other file-related operating system handles such as pipes, standard input, amd standard output. You can specify read and write operations to be either synchronous or asynchronous. FileStream buffers input and output for better performance. FileStream objects support random access to files using the Seek method. Seek allows the read/write position to be moved to any position within the file. This is done with byte offset reference point parameters. The byte offset is relative to the seek reference point, which can be the beginning, the current position, or the end of the underlying file, as represented by the three properties of the SeekOrigin class. Note Disk files always support random access. At the time of construction, the CanSeek property value is set to true or false depending on the underlying file type. Specifically, if the underlying file type is FILE\_TYPE\_DISK, as defined in winbase.h, the CanSeek property value is true. Otherwise, the CanSeek property value is false. Although the synchronous methods Read and Write and the asynchronous methods BeginRead, BeginWrite, EndRead, and EndWrite can work in either synchronous or asynchronous mode, the mode affects the performance of these methods. FileStream defaults to opening files synchronously, but provides the FileStream(IntPtr, FileAccess, Boolean, Int32, Boolean) and FileStream(String, FileMode, FileAccess, FileShare, Int32, Boolean) constructors to open files asynchronously. If a process terminates with part of a file locked or closes a file that has outstanding locks, the behavior is undefined. For directory and other file operations, see the File, Directory, and Path classes. The File class is a utility class with static methods primarily for the creation of FileStream objects based on file paths and the standard input, standard output, and standard error devices. The MemoryStream class creates a stream from a byte array and functions similarly to a FileStream. For an example of using this class, see the Example section below. The following table lists examples of other typical or related I/O tasks.

| To do this...          | See the example in this topic... |
|------------------------|----------------------------------|
| Create a text file.    | Writing Text to a File           |
| Write to a text file.  | Writing Text to a File           |
| Read from a text file. | Reading Text from a File         |

|                        |                                                                            |
|------------------------|----------------------------------------------------------------------------|
| Append text to a file. | Opening and Appending to a Log File<br>File.AppendText FileInfo.AppendText |
|------------------------|----------------------------------------------------------------------------|

|                                                   |                                                  |
|---------------------------------------------------|--------------------------------------------------|
| Rename or move a file.                            | File.Move FileInfo.MoveTo                        |
| Delete a file.                                    | File.Delete FileInfo.Delete                      |
| Copy a file.                                      | File.Copy FileInfo.CopyTo                        |
| Get the size of a file.                           | FileInfo.Length                                  |
| Get the attributes of a file.                     | File.GetAttributes                               |
| Set the attributes of a file.                     | File.SetAttributes                               |
| Determine if a file exists.                       | File.Exists                                      |
| Read from a binary file.                          | Reading and Writing to a Newly Created Data File |
| Write to a binary file.                           | Reading and Writing to a Newly Created Data File |
| Retrieve a file extension.                        | Path.GetExtension                                |
| Retrieve the fully qualified path of a file.      | Path.GetFullPath                                 |
| Retrieve the file name and extension from a path. | Path.GetFileName                                 |
| Change the extension of a file.                   | Path.ChangeExtension                             |

**Question: 34**

You are an application developer for Company.com. You are developing a Windows Forms client application that will be used within Company to access data in a Microsoft SQL Server database. The application defines a SqlConnection object named objConn and a SqlCommand object named objCmd.The application also includes the following code segment.

```
Dim objReader As SqlDataReader = obj.Cmd.ExecuteReader()
```

You need to improve the security of this code segment. You decide to replace the existing code segment.

Which code segment should you use?

- A. Try    Dim objReader As SqlDataReader = objCmd.ExecuteReader()    Catch    MsgBox “An error occurred while querying SQL Server.” End Try
- B. Dim objReader As SqlDataReader    objReader = objCmd.ExecuteReader()    objConn.Close()

```
C. Try Dim objReader As SqlDataReader objReader = objCmd.ExecuteReader() Catch e As Exception
 MsgBox e.Message
End Try
```

```
D. Dim objReader As SqlDataReader objReader = objCmd.ExecuteReader() If Not objReader.HasRows
 Then MsgBox "An error occurred while querying SQL Server." End If
```

**Answer: A**

### Explanation

You can use the ADO.NET DataReader to retrieve a read-only, forward-only stream of data from a database. Results are returned as the query executes, and are stored in the network buffer on the client until you request them using the Read method of the DataReader. Using the DataReader can increase application performance both by retrieving data as soon as it is available, rather than waiting for the entire results of the query to be returned, and (by default) storing only one row at a time in memory, reducing system overhead. After creating an instance of the Command object, you create a DataReader by calling Command.ExecuteReader to retrieve rows from a data source, as shown in the following example. Dim myReader As SqlDataReader = myCommand.ExecuteReader() You use the Read method of the DataReader object to obtain a row from the results of the query. You can access each column of the returned row by passing the name or ordinal reference of the column to the DataReader. However, for best performance, the DataReader provides a series of methods that allow you to access column values in their native data types (GetDateTime, GetDouble, GetGuid, GetInt32, and so on). For a list of typed accessor methods, see the OleDbDataReader Class and the SqlDataReader Class. Using the typed accessor methods, when the underlying data type is known, reduces the amount of type conversion required when retrieving the column value.

**Note** The Windows Server 2003 release of the .NET Framework includes an additional property for the **DataReader**, **HasRows**, which enables you to determine if the **DataReader** has returned any results before reading from it. Keep Exception Information Private Attackers often use information from an exception, such as the name of your server, database, or table to mount a specific attack on your system. Because exceptions can contain specific information about your application or data source, you can help your application and data source better protected by only exposing information to the client that is required. To avoid exposing private information through exceptions, do not return the contents of a system exception to the user. Instead, handle the exception internally. If a message must be sent to the user, return your own custom message that contains minimal information (such as "Connection failed. Please contact your system administrator."), and log the specific information so that an administrator can utilize it

### Try....Catch.....Finally Statement

When an application runs, the code in the Try block is executed first, if the code gives any error, the error is handled by the catch block. We can have multiple catch block in the Try statement. (Because, our code may raise different exception on different situation.) For example, Let us assume that we are doing both file operation and database operation, there are changes for getting the IOExceptions, File Exceptions and Database exceptions. We need to handle all the exceptions, ie., we need to be prepared for everything (Whatever be the exception) our code should be capable of handling them.

**Syntax for Try....catch statement** Try ' Block of code, where the chances of getting exception is there catch <Exception object> 'Block of code that needs to be executed when the exception of that type is caught Finally 'Block of code that needs to be executed, even when there is no exceptions. End Try Finally block is executed just before exists the Try...Catch...Finally back in which the error occurred. Finally block is executed even when the exception is not raised. Finally block is very useful. For example, we are doing some file operations, where i have opened a file, if the file is opened, it will locked for editing the file, lock is removed only when we close the file. So we need to close the file even if the exception is raised or not.

### Example for Structured Exception Handling

```
Public Function Structeddivide(ByVal a As Integer, ByVal b As Integer) As Integer Try Return a / b Catch ex As OverflowException
 MsgBox("Cannot divide by zero") Return -1 Finally MsgBox("Calculation over") End Try End Function
```

New exception from SqlDataReader. In the System.Data.SqlClient namespace, you construct a SqlDataReader with code similar to this: Dim myCommand As New SqlCommand(mySelectQuery, myConnection) myConnection.Open() Dim myReader As SqlDataReader myReader = myCommand.ExecuteReader() In 1.0 this code would succeed, even if the command was chosen as the deadlock victim by SQL Server (in the case where there's a locking issue with another connection, of course). You wouldn't get a SqlException back

until you actually tried to read data from the SqlDataReader. In

1.1 the ExecuteReader() method can now throw a SqlException. If you've got error handling that depends on only seeing this particular exception when you're actually reading data, you'll need to revise the code to move up.

### Question: 35

You are an application developer for your company, which is named Contoso, Ltd. You are developing an application that stores configuration data in a file named C:\Contoso\Persistence.config. This file is the only file your application will access. The design document for the application specifies the following two requirements: All authenticated users are allowed to view the contents of the configuration data file. Only members of a group named Managers are allowed to modify the data in the configurationdata file. You need to ensure that the file can be accessed according to these requirements.

Which two actions should you perform? (Each correct answer presents part of the solution. Choose two.)

- A. Apply a discretionary access control list (DACL) entry on the file. Use the DACL to grant Read permission to all authenticated users, and to grant Write permission to the Managers group.
- B. Apply a discretionary access control list (DACL) entry on the file. Use the DACL to grant Read permission to the Everyone group, and to grant Write permission to the Managers group.
- C. Add the following code segment to the application before accessing the file. Dim Wp As New WindowsPrincipal(WindowsIdentity.GetCurrent())Dim Manager As BooleanManager = Wp.IsInRole("Managers")Dim Fp As FileIOPermissionDim ConfigFile As StringConfigFile = "C:\Contoso\Persistence.config"If Manager Then Fp = New FileIOPermission(FileIOPermissionAccess.AllAccess, ConfigFile)Else Fp = New FileIOPermission(FileIOPermissionAccess.Read, ConfigFile)End IfFp.PermitOnly()
- D. Add the following code segment to the application before accessing the file. Dim Wp As New WindowsPrincipal(WindowsIdentity.GetCurrent())If Wp.IsInRole("Managers") Then Dim Anon As WindowsIdentity Anon = WindowsIdentity.GetAnonymous() Anon.Impersonate()End If
- E. Require users to enter a password that is shared among members of the Managers group whenever the application opens the file for writing.

**Answer: B, D**

### Question: 36

You are an application developer for Company.com. You are developing a Windows Service application. Your user account is a member of only the Users group on your computer. A written company policy states that developers are not allowed to log on by using an account that has more authority than is needed. You need to develop and debug the application. What should you do?

- A. Modify the application to run the Microsoft Visual Studio .NET debugger with the FullTrust permission by using code access security policy.
- B. Set the discretionary access control list (DACL) permissions on the executable file for the application to grant your user account Full Control permission for the executable file.
- C. Create a user account that is a member of the Debuggers Users group. Use this account for debugging.
- D. Start the development environment from the command line by running the **runas** command and specifying an account in the Administrators group.

**Answer: C**

### Explanation

Security is important. No one argues with this and everyone spends a lot of time thinking about security issues, security bugs, and malicious users. However, very few people are willing make the effort to eradicate the largest single reason that e-mail viruses and cracks in general are so dangerous: everyone logs in as a user who is a member of the local Administrators group, and most services run as Administrators. The principle of "least privilege" states that running with the minimal set of rights needed to perform an action

minimizes the damage done when something bad happens, whether it is a corrupt attachment in an e-mail received from Outlook, or a service that has a security risk. By running programs without Administrative privileges whenever possible, you ensure a more secure environment. Currently available software often requires elevated privileges in order to run correctly. To end this situation, developers must take the first step and stop running as administrators. Then, if we all consistently log in, develop, and exams applications as non-administrative users, the software we produce is more likely to be executable without artificial requirements of elevated privileges. Until developers fix the software they are writing and shipping, users will never be able to run in a secure environment, too! Both Web applications and Web services can be easily debugged with Visual Studio .NET. By default, pressing F5 within a Web services project will display a simple exams page, through which you can call the Web services you have developed. Debugging this code is simple. Just set your breakpoints in your Web service code and invoke the method through the exams page.

Your breakpoints will be hit as the Web method is invoked. You can also step into an ASP.NET Web service from an application written for the runtime. To do this, make sure you have added the account used to run ASP.NET (typically ASPNET) to the Debugger Users group on your machine. Then start debugging the Web or client application. When you reach the line that makes the call into the Web service, step into the function call using the Step Into command in the debugger. In the background, the debugger will automatically step into the Web service for you. 5 permission levels, current code sets the highest first before evaluating the other levels and only evaluates levels for Editors and Reviewers. Correct the code in accordance with the permission matrix and improve security.

| Value | Permission             | Used for group                           |
|-------|------------------------|------------------------------------------|
| 0     | None                   | All except Reviewers, Editors and Admins |
| 1     | Read Only              | Reviewers, Editors and Admins            |
| 2     | Write                  | Editors and Admins                       |
| 3     | Read and Write         | Editors and Admins                       |
| 4     | Read, Write and delete | Admins                                   |

### Question: 37

You are an application developer for Company.com. You are conducting a code review of an application that was developed by another developer. The application declares a variable named CompanyPermissions. The value of this variable indicates which permissions a user has for the application. Each value represents a specific permission or set of permissions, and user groups are permitted to have specific permissions. The permissions and groups are shown in the following table.

| Value | Permission              | Used for group                            |
|-------|-------------------------|-------------------------------------------|
| 0     | None                    | All except Reviewers, Editors, and Admins |
| 1     | Read only               | Reviewers, Editors, and Admins            |
| 2     | Write                   | Editors and Admins                        |
| 3     | Read and write          | Editors and Admins                        |
| 4     | Read, write, and delete | Admins                                    |

The application stores the current user's group name in a variable named strGroup. Each user can belong to

only one group. The application sets the value of CompanyPermissions as shown in the following code segment.

```
CompanyPermissions = 4
If strGroup = "Editors" Then CompanyPermissions -= 1
End If
If strGroup = "Reviewers" Then CompanyPermissions -= 3
End If
```

You need to improve the security of this code segment as much as possible while maintaining its functionality. You decide to replace the existing code segment. Which code segment should you use?

- A. 

```
CompanyPermissions = 4
If strGroup <> "Admins" Then CompanyPermissions -= 1
End If
If strGroup <> "Editors" Then CompanyPermissions -= 2
End If
If strGroup <> "Reviewers" Then CompanyPermissions -= 1
End If
```
- B. 

```
CompanyPermissions = 0
If strGroup = "Admins" Then CompanyPermissions = 4
End If
If strGroup = "Editors" Then CompanyPermissions = 3
End If
If strGroup = "Reviewers" Then CompanyPermissions = 1
End If
```
- C. 

```
CompanyPermissions = 1
If strGroup = "Admins" Then CompanyPermissions += 3
End If
If strGroup = "Editors" Then CompanyPermissions += 2
End If
```
- D. 

```
CompanyPermissions = 1
Select Case strGroup
Case "Admins" CompanyPermissions = 4
Case "Editors"
```

**Answer: B**

### Explanation

All access must be denied by default to ensure that permissions are evaluated appropriately. Explicit deny followed by evaluated allows. Assignment Operators: =, +=, -=, \*=, /=, %=, >>=, <<=, &=, ^=, |=  
*assignment-expression : conditional-expression unary-expression assignment-operator assignment-expression*

An assignment operation assigns the value of the right-hand operand to the storage location named by the lefthand operand.

Therefore, the left-hand operand of an assignment operation must be a modifiable l-value. After the assignment, an assignment expression has the value of the left operand but is not an l-value. The *assignment-operator* is one of the following:

| Operator | Operation Performed       |
|----------|---------------------------|
| =        | Simple assignment         |
| *=       | Multiplication assignment |
| /=       | Division assignment       |
| %=       | Remainder assignment      |
| +=       | Addition assignment       |
| -=       | Subtraction assignment    |
| <<=      | Left-shift assignment     |
| >>=      | Right-shift assignment    |

|    |                                 |
|----|---------------------------------|
| &= | Bitwise-AND assignment          |
| =  | Bitwise-inclusive-OR assignment |
| ^= | Bitwise-exclusive-OR assignment |

**Example** In the following example, the addition assignment operator (+=) is used to add 3 to nNumber: // Example of the addition assignment operator  
int nNumber=1; nNumber += 3; // nNumber now is 4

**Question: 38**

You are an application developer for Company, You develop an ASP.NET application that uses a database to keep track of hours worked by each employee. The application stores the account name of the interactive user in a variable named CompanyName. The application uses the value of CompanyName and data entered by each user to record the user name and hours worked.

The application is configured to use Integrated Windows authentication in IIS. The Web.config file has Windows authentication configured and impersonation enabled. During a security review, you find out that the application is running under a user context that has more permissions than necessary.

You need to increase the security of the application while maintaining current functionality. What should you do?

- A. Ask a network administrator to enable basic authentication for the application in IIS and prompt the user to enter the user's user name and password.
- B. Ask a network administrator to enable digest authentication for the application in IIS and prompt the user to enter the user's user name and password.
- C. Change the Web.config file to set impersonation to **false**. Add the following code to populate the CompanyName variable with the user name of the interactive user.  
Dim CompanyName As String  
CompanyName = HttpContext.Current.User.Identity.Name.ToString()
- D. Change the Web.config file to set impersonation to **false**. Add the following code to populate the CompanyName variable with the user name of the Interactive user.  
Dim userIdentity As WindowsIdentity

**Answer: C**

**Explanation**

Principal objects implement the IPrincipal interface and represent the security context of the user on whose behalf the code is running. The principal object includes the user's identity (as a contained Identity object) and any roles to which the user belongs. ASP.NET provides the following principal and identity object implementations:

WindowsPrincipal and WindowsIdentity objects represent users who have been authenticated with Windows authentication. With these objects, the role list is automatically obtained from the set of Windows groups to which the Windows user belongs.

GenericPrincipal and GenericIdentity objects represent users who have been authenticated using Forms authentication or other custom authentication mechanisms. With these objects, the role list is obtained in a custom manner, typically from a database.

FormsIdentity and PassportIdentity objects represent users who have been authenticated with Forms and Passport authentication respectively. The following tables illustrate, for a range of IIS authentication settings, the resultant identity that is obtained from each of the variables that maintain an IPrincipal and/or Identity object. The following abbreviations are used in the table:

HttpContext = HttpContext.Current.User, which returns an IPrincipal object that contains security information for the current Web request. This is the authenticated Web client.

WindowsIdentity = WindowsIdentity.GetCurrent(), which returns the identity of the security context of the currently executing Win32 thread.

Thread = Thread.CurrentPrincipal which returns the principal of the currently executing .NET thread which rides on top of the Win32 thread.

**Table 1. IIS anonymous authentication**

| Web.config Settings                                                  | Variable Location                  | Resultant Identity                                                     |
|----------------------------------------------------------------------|------------------------------------|------------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread | -MACHINE\IUSR_MACHINE<br>-                                             |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread | -MACHINE\ASPNET -                                                      |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\IUSR_MACHINE<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET Name<br>provided by user       |

**Table 2. IIS basic authentication**

| Web.config Settings                                                  | Variable Location                  | Resultant Identity                                                |
|----------------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread | Domain\UserName<br>Domain\UserName<br>Domain\UserName             |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread | Domain\UserName<br>MACHINE\ASPNET<br>Domain\UserName              |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread | Name provided by user<br>Domain\UserName<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user  |

**Table 3. IIS digest authentication**

| Web.config Settings | Variable Location | Resultant Identity |
|---------------------|-------------------|--------------------|
|---------------------|-------------------|--------------------|

|                                                                      |                                    |                                                                   |
|----------------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread | Domain\UserName<br>Domain\UserName<br>Domain\UserName             |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread | Domain\UserName<br>MACHINE\ASPNET<br>Domain\UserName              |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread | Name provided by user<br>Domain\UserName<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user  |

**Table 4: IIS integrated Windows**

| Web.config Settings                                                  | Variable Location                   | Resultant Identity                                                |
|----------------------------------------------------------------------|-------------------------------------|-------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread  | Domain\UserName<br>Domain\UserName<br>Domain\UserName             |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread  | Domain\UserName<br>MACHINE\ASPNET<br>Domain\UserName              |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread  | Name provided by user<br>Domain\UserName<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext. WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user  |

HttpContext.User Property Gets or sets security information for the current HTTP request. Public Property User As IPincipal Property Value Security information for the current HTTP request. Remarks Setting this property requires the ControlPrincipal flag to be set in Flags. The HttpContext.User property provides programmatic access to the properties and methods of the IPincipal interface. Because ASP.NET pages contain a default reference to the System.Web namespace (which contains the HttpContext class), you can reference the members of HttpContext on an .aspx page without the fully qualified class reference to HttpContext. For example, you can use just User.Identity.Name to get the name of the user on whose behalf the current process is running. If you want to use the members of IPincipal from an ASP.NET code-behind module, however, you must include a reference to the System.Web namespace in the module and also fully qualify the reference to

the currently active request/response context and the class in System.Web you want to use. For example, in a code-behind page you must specify the full name HttpContext.Current.User.Identity.Name.

### Question: 39

You are an application developer for your company. You are conducting a code review of an application that updates a Microsoft SQL Server database named Payroll. This database is used by other applications. The application contains the following code segment. Public Sub calculateAndStore(ByVal Conn As SqlConnection, \_ ByVal ID As String, ByVal Bonus As String, ByVal Salary As String) Salary = Convert.ToString(Convert.ToDecimal(Salary) \* 1.05) Bonus = Convert.ToString(Convert.ToDecimal(Salary) \* 0.05) If ID.Length = 0 Then Throw New ApplicationException("Error - Empty ID") End If Dim NewID As String NewID = "ID-" & ID Dim StrUpdate As String StrUpdate = "UPDATE Payroll SET EmployeeID = " & NewID & ", Bonus = " & Bonus & ", Salary = " & Salary & " " & \_ "WHERE EmployeeID=" & ID & "" Dim Cmd As New SqlCommand(StrUpdate, Conn) Cmd.Connection.Open() Cmd.ExecuteNonQuery() End Sub The values in the string variables named ID, Bonus, and Salary are contained in the Payroll database. The purpose of the code segment is to calculate new values for ID, Bonus, and Salary, and to update those values in the Payroll database. You need to improve the security of this application. What should you do?

- A. Validate the new Salary value is within the range for the data type in the SQL Server database.
- B. Validate the contents of the ID value before updating it in the SQL Server database.
- C. Validate the length of the ID value before updating it in the SQL Server database.
- D. Enclose the body of the function with a **Try-Catch** block.

**Answer: B**

### Question: 40

You are an application developer for your company. You are developing an application that includes administrative features that could destroy important information if used incorrectly. You need to ensure that only members of the Administrators group can use or discover the administrative features. You need to achieve this goal while minimizing the impact on users.

Which two actions should you perform? (Each correct answer presents part of the solution. Choose two.)

- A. Disable the menu choices for all administrative features if the user is not a member of the Administrators group.
- B. Remove the menu choices for all administrative features if the user is not a member of the Administrators group.
- C. Throw an exception at the start of each administrative feature if the user is not a member of the Administrators group.
- D. Require users to provide an administrator password before each execution of administrative features in the application.

**Answer: C, D**

### Question: 41

You are an application developer for your company. You are conducting a code review of an assembly written by another developer. The assembly is named MyAssembly.exe. The assembly is for an application that accesses data in a Microsoft SQL Server database. All users of the application have access to the database by using their Microsoft Windows user accounts. The assembly contains the following code segment. Dim userid As String Dim password As String userid = "sa" password = "" Dim sqlConnection As New SqlConnection Dim connectionString As String connectionString = "data source=myserver" connectionString &= ";initial catalog=myDatabase" connectionString &= ";user id=" & userid connectionString &= ";password=" & password sqlConnection.ConnectionString = connectionString sqlConnection.Open()

You need to improve the security of the code segment. What should you do?

- A. Replace the code segment with the following code segment. Dim sqlConnection As New SqlConnectionDim connectionString As StringconnectionString = "data source=myserver"connectionString &= ";Integrated Security=SSPI"connectionString &= ";initial catalog=myDatabase"sqlConnection.ConnectionString = connectionStringsqlConnection.Open()
- B. Replace the code segment with the following code segment. Dim sqlConnection As SqlConnectionDim connectionString As StringconnectionString = \_ "data source=myserver;initial catalog=myDatabase;user id=sa;password=;"sqlConnection.ConnectionString = connectionStringsqlConnection.Open()
- C. Run the caspol.exe -resolveperm MyAssembly.exe command from the command line.
- D. Run the permview /decl MyAssembly.exe command from the command line.

**Answer: A**

#### **Question: 42**

You are an application developer for your company. You are developing a Windows Forms application. You deploy a supporting assembly named MyAssembly.dll to the global assembly cache. During testing, you discover that the application is prevented from accessing MyAssembly.dll. You need to ensure that the application can access MyAssembly.dll. What should you do?

- A. Digitally sign the application by using a digital certificate.
- B. Run the caspol.exe -s on command from the command line.
- C. Run the Assembly Linker to link MyAssembly.dll to the application.
- D. Modify the security policy to grant the application the FullTrust permission.

**Answer: D**

#### **Question: 43**

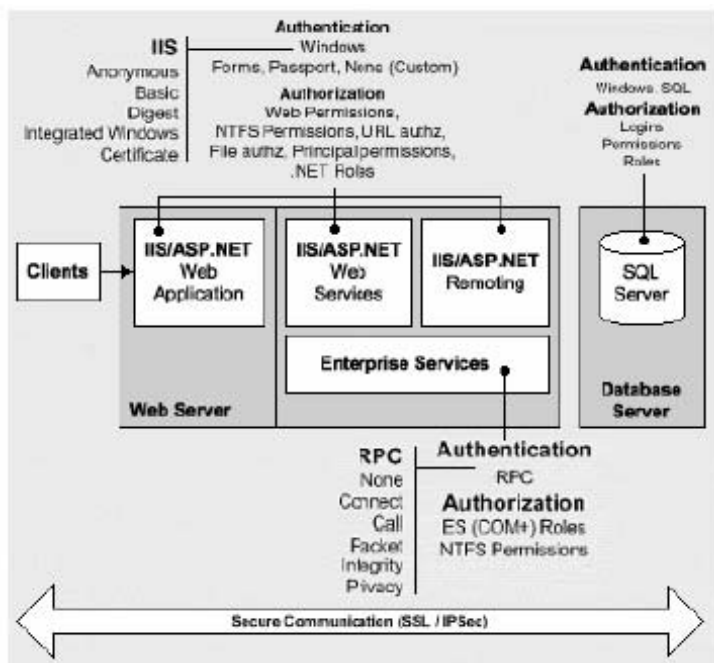
You are an application developer for your company. You develop an ASP.NET Web application that is installed on a server named Server1. Server1 has IIS 5.0 installed. The Web application is configured to use Anonymous authentication in IIS. The Web.config file contains the following code segment. <authentication mode="Windows" /> <identity impersonate="true" /> The Machine.config file contains the following code segment. <authentication mode="Windows" /> The application implements security based on the following code segment. Dim myIdentity As String Dim validAccess As IntegervalidAccess = 0 myIdentity = WindowsIdentity.GetCurrent().Name.ToString() If (myIdentity = "Server1\ASPNET") Then validAccess = 1 End If For testing purposes, you display the value of the validAccess variable in a label. When you run the application, you discover that the value of validAccess is 0. You need to ensure that validAccess has a value of 1. What should you do?

- A. Replace the code segment in the Web.config file with the following code segment. <authentication mode="Windows" /><identity impersonate="false" />
- B. Replace the code segment in the Web.config file with the following code segment. <authentication mode="Forms" /><identity impersonate="true" />
- C. Ask a network administrator to change the authentication mode of the Web application to Integrated Windows authentication.
- D. Ask a network administrator to change the authentication mode of the Web application to basic authentication.

**Answer: A**

#### **Explanation**

Microsoft's Security Architecture provides for 5 Authentication modes in IIS and 4 Authentication modes in ASP.NET These include Anonymous, Basic,Digest,Windows Integrated (Kerberos/NTLM) and Certificate authentications for IIS and None, Windows, Forms and Passport for ASP.NET



For each of the authentication modes in ASP.NET, 'Impersonate' can be set to False or True. Setting Impersonate to False results in ASP.NET applications using the default system account ASPNET to access Resources. Setting Impersonate to True results in ASP.NET using the identity passed by IIS(IUSR\_machinename , domain\username , etc).

- \* HttpContext = This is the authenticated Web client.
  - \* WindowsIdentity = This is the identity of the security context of the currently executing Win32 thread.
  - \* Thread = This is the identity of the currently executing .NET thread which rides on top of the Win32 thread
- Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication Principal objects implement the IPrincipal interface and represent the security context of the user on whose behalf the code is running. The principal object includes the user's identity (as a contained Identity object) and any roles to which the user belongs. ASP.NET provides the following principal and identity object implementations:

WindowsPrincipal and WindowsIdentity objects represent users who have been authenticated with Windows authentication. With these objects, the role list is automatically obtained from the set of Windows groups to which the Windows user belongs.

**GenericPrincipal** and **GenericIdentity** objects represent users who have been authenticated using Forms authentication or other custom authentication mechanisms. With these objects, the role list is obtained in a custom manner, typically from a database.

**FormsIdentity** and **PassportIdentity** objects represent users who have been authenticated with Forms and Passport authentication respectively. The following tables illustrate, for a range of IIS authentication settings, the resultant identity that is obtained from each of the variables that maintain an IPrincipal and/or Identity object. The following abbreviations are used in the table:

HttpContext = HttpContext.Current.User, which returns an IPrincipal object that contains security information for the current Web request. This is the authenticated Web client.

WindowsIdentity = WindowsIdentity.GetCurrent(), which returns the identity of the security context of the currently executing Win32 thread.

Thread = Thread.CurrentPrincipal which returns the principal of the currently executing .NET thread which rides on top of the Win32 thread.

**Table 1. IIS anonymous authentication**

| Web.config Settings | Variable Location | Resultant Identity |
|---------------------|-------------------|--------------------|
|---------------------|-------------------|--------------------|

|                                                                      |                                    |                                                                        |
|----------------------------------------------------------------------|------------------------------------|------------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread | -MACHINE\IUSR_MACHINE -                                                |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread | -MACHINE\ASPNET -                                                      |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\IUSR_MACHINE<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user       |

**Table 2. IIS basic authentication**

| Web.config Settings                                                  | Variable Location                  | Resultant Identity                                                |
|----------------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread | Domain\UserName<br>Domain\UserName<br>Domain\UserName             |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread | Domain\UserName<br>MACHINE\ASPNET<br>Domain\UserName              |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread | Name provided by user<br>Domain\UserName<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user  |

**Table 3. IIS digest authentication**

| Web.config Settings                                                  | Variable Location                  | Resultant Identity                                    |
|----------------------------------------------------------------------|------------------------------------|-------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread | Domain\UserName<br>Domain\UserName<br>Domain\UserName |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread | Domain\UserName<br>MACHINE\ASPNET<br>Domain\UserName  |

|                                                                    |                                    |                                                                   |
|--------------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Forms" />  | HttpContext WindowsIdentity Thread | Name provided by user<br>Domain\UserName<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" /> | HttpContext WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user  |

**Table 4: IIS integrated Windows**

| Web.config Settings                                                  | Variable Location                   | Resultant Identity                                                |
|----------------------------------------------------------------------|-------------------------------------|-------------------------------------------------------------------|
| <identity impersonate="true"/><br><authentication mode="Windows" />  | HttpContext WindowsIdentity Thread  | Domain\UserName<br>Domain\UserName<br>Domain\UserName             |
| <identity impersonate="false"/><br><authentication mode="Windows" /> | HttpContext WindowsIdentity Thread  | Domain\UserName<br>MACHINE\ASPNET<br>Domain\UserName              |
| <identity impersonate="true"/><br><authentication mode="Forms" />    | HttpContext WindowsIdentity Thread  | Name provided by user<br>Domain\UserName<br>Name provided by user |
| <identity impersonate="false"/><br><authentication mode="Forms" />   | HttpContext. WindowsIdentity Thread | Name provided by user<br>MACHINE\ASPNET<br>Name provided by user  |

The machine configuration file, Machine.config, contains settings that apply to an entire computer. This file is located in the %runtime install path%\Config directory. Machine.config contains configuration settings for machine-wide assembly binding, built-in remoting channels, and ASP.NET. The configuration system first looks in the machine configuration file for the <appSettings> element and other configuration sections that a developer might define. It then looks in the application configuration file. To keep the machine configuration file manageable, it is best to put these settings in the application configuration file. However, putting the settings in the machine configuration file can make your system more maintainable. For example, if you have a third-party component that both your client and server application uses, it is easier to put the settings for that component in one place. In this case, the machine configuration file is the appropriate place for the settings, so you don't have the same settings in two different files

ASP.NET configuration, of which security is a part, has a hierarchical architecture. All configuration information for ASP.NET is contained in files named Web.config and Machine.config. Web.config can be placed in the same directories as the application files. The Machine.config file is in the Config directory of the install root. Subdirectories inherit a directory's settings unless overridden by a Web.config file in the subdirectory. In a Web.config file, there are sections for each major category of ASP.NET functionality.

**Question: 44**

You are an application developer for Company.com. You develop an application that uses an external class library. You run the Permissions View tool on the class library and receive the following output. Microsoft (R) .NET Framework Permission Requests Viewer. Version 1.1.4322.753 Copyright (C) Microsoft Corporation 1998-2002. All rights reserved. minimal permission set: <PermissionSet class="System.Security.PermissionSet" version="1"> <IPermission class="System.Security.Permission.ReflectionPermission, mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Flags="ReflectionEmit"/> <IPermission class="System.Security.Permissions.SecurityPermission, mscorlib, Version=1.0.5000.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" version="1" Flags="SerializationFormatter"/> </PermissionSet> optional permission set: <PermissionSet class="System.Security.PermissionSet" version="1" Unrestricted="true"/> refused permission set: Not specified

You need to add corresponding attributes in your application.

Which code segment should you use?

- A. <Assembly: ReflectionPermission(SecurityAction.RequestRefuse, \_ ReflectionEmit:=False), \_ Assembly: SecurityPermission(SecurityAction.RequestRefuse, \_ SerializationFormatter:=False), \_ Assembly: PermissionSetAttribute(SecurityAction.RequestOptional, Unrestricted:=True)>
- B. <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, \_ ReflectionEmit:=False), \_ Assembly: SecurityPermission(SecurityAction.RequestRefuse, \_ SerializationFormatter:=False), \_ Assembly: PermissionSetAttribute(SecurityAction.RequestRefuse, Unrestricted:=True)>
- C. <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, \_ ReflectionEmit:=False), \_ Assembly: SecurityPermission(SecurityAction.RequestMinimum, \_ SerializationFormatter:=False), \_ Assembly: PermissionSetAttribute(SecurityAction.RequestOptional, Unrestricted:=True)>
- D. <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, \_ ReflectionEmit:=True), \_ Assembly: SecurityPermission(SecurityAction.RequestMinimum, \_ SerializationFormatter:=True), \_ Assembly: PermissionSetAttribute(SecurityAction.RequestOptional, Unrestricted:=True)>

**Answer: D**

### Explanation

Reflection emit is a runtime feature that allows code to create dynamic assemblies, modules, and types. You can dynamically create instances of these types to use, or you can use reflection emit to generate an assembly and persist it to disk as an .exe file or DLL. Since you do not necessarily have control over what permissions are assigned to the code you write, the common language runtime provides a mechanism for requesting the permissions that you feel your code must have in order to run properly. If the code is not granted the required permissions, it will not run. And, because permission requests are stored in an assembly's manifest, the end user can run a tool to determine what permissions have been requested by the assembly author and then take the appropriate steps to grant those permissions if they need the code to run on their machine. Three types of permission requests are supported: **RequestMinimum:** The permissions the code must have to run properly. If these permissions cannot be granted, the code will not be executed. **RequestOptional:** The permissions that should be granted if allowed by policy. The runtime will attempt to execute code even if permissions it requests as optional have not been granted. **RequestRefuse:** The permissions that code should never be granted. Code will not receive these permissions, even if they would normally be granted to it. This is an extra precaution you can take to prevent your code from being misused. Permission requests can only be made in a declarative fashion and must always be at the assembly level (the assembly is the unit to which permissions are granted by the security system). The following code is a

request stating that an assembly must have unrestricted access to the file system in order to function.

```
[assembly:FileIOPermission(SecurityAction.RequestMinimum, Unrestricted=true)] public class FileMover { //something interesting }
<Assembly: FileIOPermission(SecurityAction.RequestMinimum, Unrestricted := True)> Public Class FileMover 'something interesting
End Class
```

Several requests of the same type can be made, in which case the final permission set requested is the aggregate of all requests of that type. In the example below RequestMinimum is used twice with different permissions to state that the assembly must have the ability to use Reflection Emit and perform serialization in order for it to function.

```
[assembly:ReflectionPermission(SecurityAction.RequestMinimum, ReflectionEmit=true)]
```

```
[assembly:SecurityPermission(SecurityAction.RequestMinimum, SerializationFormatter=true)] public class CodeGenerator {
//something interesting } <Assembly: ReflectionPermission(SecurityAction.RequestMinimum, ReflectionEmit := True)> <Assembly:
SecurityPermission(SecurityAction.RequestMinimum, SerializationFormatter := True)> Public Class CodeGenerator 'something
interesting End Class
```

The same permission can also appear in requests of different types. For instance, the example program at the bottom of this page uses an EnvironmentPermission in each of its three requests (Minimum, Optional, and\ Refuse). This is useful when a permission encompasses a number of operations and you want to ensure that your assembly has the ability to perform some of those operations while being prevented from performing others. It is important to note that any permission you refuse using RequestRefuse will not be granted to your assembly even if you request that same permission using RequestMinimum. In addition to requesting individual permissions, entire sets of permissions can be requested in a compact fashion. The example below shows two requests: one stating that an assembly must have unrestricted access to the file system in order to function and one stating that it will take any and all other permissions that the security system is willing to grant it.

```
[assembly:FileIOPermission(SecurityAction.RequestMinimum, Unrestricted=true)]
```

```
[assembly:PermissionSet(SecurityAction.RequestOptional, Name="FullTrust")] public class FileMover { //something interesting }
```

```
<Assembly: FileIOPermission(SecurityAction.RequestMinimum, Unrestricted := True)> <Assembly:
PermissionSet(SecurityAction.RequestOptional, Name := "FullTrust")> Public Class FileMover 'something interesting End Class
```

The previous example shows how to request a permission set by name, but it is also possible to use a custom permission set representing the exact permissions you want. For more information on how to do this, search for PermissionSetAttribute in the .NET Framework SDK Reference. The SDK provides a tool called PERMVIEW that is useful for verifying that your permission requests are correct. Running PERMVIEW on a compiled assembly will read the permission requests out of the assembly's manifest and display them as shown below.

**C:\>permview filemover.exe**

Microsoft (R) .NET Framework Permission Request Viewer. Version 1.0.XXXX.0 Copyright (C) Microsoft Corp.  
1998-2001

minimal permission set: <PermissionSet class="System.Security.PermissionSet" version="1"> <IPermission  
class="System.Security.Permissions.FileIOPermission" version="1" Unrestricted="true"/></PermissionSet>optional permission set:  
<PermissionSet class="System.Security.PermissionSet" version="1" Unrestricted="true"/>refused permission set: Not specified

#### Question: 45

You are an application developer for your company. You are conducting a code review of an application that was developed by another developer. The application accesses data that is stored in a Microsoft SQL Server database. The application contains the following code segment, which encrypts a SQL Server connectionstring. Public Function EncryptData(ByVal strConn As String) As Byte() Dim data As Byte() = Encoding.UTF8.GetBytes(strConn) Dim csp As New DESCryptoServiceProvider Dim ms As New MemoryStream ms.Write(csp.Key, 0, csp.Key.Length) ms.Write(csp.IV, 0, csp.IV.Length) Dim cs As New CryptoStream(ms, csp.CreateEncryptor(), CryptoStreamMode.Write) cs.Write(data, 0, data.Length) cs.FlushFinalBlock() Return ms.ToArray() End Function You need to improve the data encryption in the application. Which action or actions should you perform? (Choose all that apply.)

- A. Add zeros to pad the data array to a multiple of the BlockSize property of the csp variable before encrypting.
- B. Encrypt the csp.Key property and the csp.IV property by using the user's public key before storing them in the MemoryStream object named ms.
- C. Write the csp.Key property and the csp.IV property to the CryptoStream object named cs instead of the MemoryStream object named ms.
- D. Select a longer key than the default key length for the DESCryptoServiceProvider class.

**Answer: B, D**

**Question: 46**

You are an application developer for Company.com. You are implementing an ASP.NET Web application that uses Forms authentication. User names and passwords are stored in a Microsoft SQL Server database. The application includes the following method, which returns a value of True if the user provides a user name and password that are found in the database.

```
Private Function VerifyPassword(ByVal userName As String, _
 ByVal password As String) As Boolean
 You configure your application to redirect unauthenticated requests to a page named Logon.aspx. This page includes text boxes for entering a user name and password, and includes a Logon button.
```

You need to write the code to authenticate a user.

What should you do?

- A. Add the following code to the Click event handler for the **Logon** button. If VerifyPassword(txtUserName.Text, txtPassword.Text) Then Dim authTicket As FormsAuthenticationTicket = New \_ FormsAuthenticationTicket(txtUserName.Text, False, 30) Dim encTicket As String = FormsAuthentication.Encrypt(authTicket) Dim authCookie As HttpCookie = \_
- B. Add the following code to the Click event handler for the **Logon** button. If VerifyPassword(txtUserName.Text, txtPassword.Text) Then FormsAuthentication.Authenticate(txtUserName.Text,txtPassword.Text) FormsAuthentication.RedirectFromLoginPage(txtUserName.Text, False) End If
- C. Add the following code to the Application\_OnAuthenticate event handler. Dim userName As String = Context.Session("UserName") Dim password As String = Context.Session("Password") If VerifyPassword(userName, password) Then Dim authTicket As FormsAuthenticationTicket = New \_ FormsAuthenticationTicket(userName, False, 30) Dim encTicket As String = FormsAuthentication.Encrypt(authTicket) Dim authCookie As HttpCookie = \_ New HttpCookie(FormsAuthentication.FormsCookieName, encTicket) Response.Cookies.Add(authCookie) FormsAuthentication.RedirectFromLoginPage(userName, False) End If
- D. Add the following code to the Application\_OnAuthenticate event handler. Dim userName As String = Context.Session("UserName") Dim password As String = Context.Session("Password")

**Answer: A**

**Explanation**

FormsAuthentication.RedirectFromLoginPage Method (String, Boolean) Redirects an authenticated user back to the originally requested URL. Overloads Public Shared Sub RedirectFromLoginPage( \_ ByVal *userName* As String, \_ ByVal *createPersistentCookie* As Boolean \_

A. Parameters

*userName*  
Name of the user for cookie authentication purposes. This does not need to map to an account name and will be used by URL Authorization. *createPersistentCookie* Specifies whether or not a durable cookie (one that is saved across browser sessions) should be issued. Remarks The RedirectFromLoginPage method redirects to the return URL key specified in the query string. For example, in the URL http://www.contoso.com/login.aspx?ReturnUrl=caller.aspx, caller.aspx is the return URL that RedirectFromLoginPage redirects to. If the return key does not exist, RedirectFromLoginPage redirects to Default.aspx. ASP.NET automatically adds the return URL when the browser is redirected to the login page specified in the loginUrl attribute in the <forms> Element configuration directive. The method issues an authentication ticket and does a SetForms with the ticket, using the appropriately configured cookie name for the application as part of the redirect response.

**FormsAuthenticationTicket Class**

Provides a means of creating and reading the values of a forms authentication cookie (containing an authentication ticket) as used by FormsAuthenticationModule. This class cannot be inherited

**Question: 47** You are an application developer for Company.com. You are modifying an existing communications application so that the application can be used on the Internet.

You need to enhance the security of data when it is transmitted by using the application. You want to achieve this goal by using the minimum amount of development effort.

What should you do?

- A. Use HTTPS for all exchange of data.
- B. Encrypt the data by using the sender's public key and the recipient's private key.
- C. Encrypt the data by using the sender's private key and the recipient's public key.
- D. Create a key to encrypt data for each exchange between the sender and recipient.    Share the random key with the recipient.    Encrypt subsequent data by using the shared random key.
- E. Create a random key to encrypt data for each exchange between the sender and recipient.    Encrypt the random key by using the recipient's public key.    Send the encrypted random key to the recipient.    Encrypt subsequent data by using the shared random key.

**Answer: A**

### **Explanation**

HTTPS Offers:

*Client-server, end-to-end encryption*

All the HTTP traffic between the client and the server is encrypted, preventing anyone from understanding it even if they can intercept it.

### **Message Integrity**

Integrity checks ensure that the messages making up the HTTP traffic cannot be altered in transit, neither can messages be added or removed from the sequence, without detection.

### **Strong authentication of the server**

Simply providing encryption and message integrity gives little security if you don't know who the other party in the conversation actually is. With plain HTTP, your only assurance is that your browser has probably connected to the host whose name appeared in the URL you followed. This may not be the case (for example it is easy to subvert the name-to-address mapping process), and in any case it is difficult to tell who is actually operating the server that responds to any particular name. It is also fairly easy to mount a 'man in the middle' attack against plain HTTP. Under HTTPS, all servers offer the browser a cryptographic 'certificate'. These certificates are issued by trusted third parties and contain information that identifies the server and the organization operating it.

### **Optional authentication of the browser user**

Optionally, HTTPS also allows the browser to supply a certificate to the server. This can provide strong authentication of the identity of the browser user, but this feature is rarely used, probably because of the difficulty of issuing such certificates to all users. Certificates are also large, making it difficult for mobile users to have them to hand when needed. Embedding certificates in portable tokens is one approach to solving these problems.

### **Question: 48**

You are an application developer for Company.com. You are developing a Windows Forms application. Users will run your application from a Web folder on the intranet. The application stores configuration information in isolated storage. The application will read from the registry if it has the appropriate permission, but the application can run successfully without this permission.

You add the following attribute to the application.

```
<Assembly: RegistryPermission(SecurityAction.RequestOptional, _
All:="HKEY_CURRENT_USER\Software\Company\LastRun")>
```

When you run the application from the intranet, a SecurityException is thrown.

You need to modify attributes to indicate the application's exact permission requirements and correct the problem that is causing the SecurityException exception.

What should you do?

- A. Add the following attributes to the assembly. <Assembly: IsolatedStorageFilePermission(SecurityAction.RequestMinimum, \_ UsageAllowed:=IsolatedStorageContainment.AssemblyIsolationByUser, \_ UserQuota:=9223372036854775897)> Make no change to the RegistryPermission attribute.
- B. Add the following attributes to the assembly. <Assembly: IsolatedStorageFilePermission(SecurityAction.RequestMinimum, \_ UsageAllowed:=IsolatedStorageContainment.AssemblyIsolationByUser, \_ UserQuota:=9223372036854775807)> <Assembly: UIPermission(SecurityAction.RequestMinimum)> Replace the RegistryPermission attribute with the following attribute. <Assembly: RegistryPermission(SecurityAction.RequestRefuse, \_ All:="HKEY\_CURRENT\_USER\Software\Company\LastRun")>
- C. Remove the RegistryPermission attribute and add the following attribute to the assembly. <Assembly: PermissionSet(SecurityAction.RequestMinimum, Name:="FullTrust")>
- D. Remove the RegistryPermission attribute and add no new attributes to the assembly.

**Answer: A**

### Question: 49

You are an application developer for Company.com. Company sells products to business partners over the Internet. Partners sometimes claim that they did not place an order that the company fulfilled.

You need to develop a solution that can be used to prove that a partner's orders were approved by that partner. You want your solution to be as secure as possible.

What should you do?

- A. Require each partner to identify itself by using an account name.
- B. Require all orders to be placed at an HTTPS Web site that uses Company's certificate.
- C. Require partners to digitally sign each order by using a certificate.
- D. Require partners to send an e-mail message to confirm each order.

**Answer: C Explanation**

A digital signature A means for originators of a message, file, or other digitally encoded information to bind their identity to the information. The process of digitally signing information entails transforming the information, as well as some secret information held by the sender, into a tag called a *signature*. Digital signatures are used in public key environments, and they provide nonrepudiation and integrity services. It is a way to ensure the integrity and origin of data. A digital signature provides strong evidence that the data has not been altered since it was signed and it confirms the identity of the person or entity who signed the data. This enables the important security features of *integrity* and *nonrepudiation*, which are essential for secure electronic commerce transactions. Digital signatures are typically used when data is distributed in plaintext, or unencrypted form. In these cases, while the sensitivity of the message itself may not warrant encryption, there could be a compelling reason to ensure that the data is in its original form and has not been sent by an impostor because, in a distributed computing environment, plaintext can conceivably be read or altered by anyone on the network with the proper access, whether authorized or not. nonrepudiation:

- 1 The capability, in security systems, that guarantees that a message or data can be proven to have originated from a specific person.
- 2 Assurance the sender of data is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the data. Nonrepudiation Nonrepudiation is a method of proving either that a user performed an action (such as enrolling in a stock plan or applying for a car loan), or that the user sent or received some information at a particular time. This prevents the individual from fraudulently renegeing on a transaction. By comparison, if you purchase an item, you might have to sign for the item upon receipt. If you decide to renege on the deal, the vendor can simply show you the signed receipt. A comprehensive nonrepudiation plan usually requires authentication, authorization, data integrity, and auditing. In addition, nonrepudiation requires a message on the Web page, warning that the action the user is about to take is legally binding. This does not make the Web server more secure, but it does make Web transactions (such as purchasing an item) more secure. Today, electronic nonrepudiation across the Internet is new and there is little in the way of legal precedent. This s sure to change as more business is performed across the Web.

**Question: 50**

You are an application developer for Company.com. You are developing an application that consists of a single Windows Forms assembly. This application accesses a Microsoft SQL Server database. Access to this database is critical to the successful operation of your application. This application will be distributed throughout the company. If the code access security policy prevents the application from running, an administrator must reconfigure the code access security policy to grant your assembly the required permissions for your application to run. You need to update the application so that an administrator can ascertain the permission requirements for the assembly. Which attribute should you add to the assembly?

- A. <Assembly: SqlClientPermission(SecurityAction.RequestMinimum)>
- B. <Assembly: SqlClientPermission(SecurityAction.RequestOptional)>
- C. <Assembly: AssemblyDescription("SqlClientPermission")>
- D. <Assembly: AssemblyConfiguration("SqlClientPermission")>

**Answer: A**

**Explanation**

SqlClientPermission Class Enables the .NET Framework Data Provider for SQL Server to help ensure that a user has a security level adequate to access a data source. or a list of all members of this type, see SqlClientPermission Members. System.Object System.Security.CodeAccessPermission System.Data.Common.DbDataPermission System.Data.SqlClient.SqlClientPermission <Serializable> NotInheritable Public Class SqlClientPermission Inherits DbDataPermission

**Remarks**

The IsUnrestricted property takes precedence over the AllowBlankPassword property. Therefore, if you set **AllowBlankPassword** to **false**, you must also set **IsUnrestricted** to **false** to prevent a user from making a connection using a blank password.

**SecurityAction Enumeration**

Specifies the security actions that can be performed using declarative security. <Serializable>Public Enum SecurityAction

**Remarks**

The following table describes the time that each of the security actions takes place and the targets that each supports.

| Declaration of security action | Time of action           | Targets supported |
|--------------------------------|--------------------------|-------------------|
| LinkDemand                     | Just-in-time compilation | Class, Method     |
| InheritanceDemand              | Load time                | Class, Method     |
| Demand                         | Run time                 | Class, Method     |
| Assert                         | Run time                 | Class, Method     |
| Deny                           | Run time                 | Class, Method     |
| PermitOnly                     | Run time                 | Class, Method     |
| RequestMinimum                 | Grant time               | Assembly          |
| RequestOptional                | Grant time               | Assembly          |
| RequestRefuse                  | Grant time               | Assembly          |

For additional information about attribute targets, see Attribute.

**Members**

| Member name | Description |
|-------------|-------------|
|-------------|-------------|

|                   |                                                                                                                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Assert            | The calling code can access the resource identified by the current permission object, even if callers higher in the stack have not been granted permission to access the resource (see Assert).           |
| Demand            | All callers higher in the call stack are required to have been granted the permission specified by the current permission object (see Security Demands).                                                  |
| Deny              | The ability to access the resource specified by the current permission object is denied to callers, even if they have been granted permission to access it (see Deny).                                    |
| InheritanceDemand | The derived class inheriting the class or overriding a method is required to have been granted the specified permission.                                                                                  |
| LinkDemand        | The immediate caller is required to have been granted the specified permission. For more information on declarative security and link demands, see Declarative Security Used with Class and Member Scope. |
| PermitOnly        | Only the resources specified by this permission object can be accessed, even if the code has been granted permission to access other resources (see PermitOnly).                                          |
| RequestMinimum    | The request for the minimum permissions required for code to run. This action can only be used within the scope of the assembly.                                                                          |
| RequestOptional   | The request for additional permissions that are optional (not required to run). This action can only be used within the scope of the assembly.                                                            |
| RequestRefuse     | The request that permissions that might be misused will not be granted to the calling code. This action can only be used within the scope of the assembly.                                                |

**Question: 51**

You are an application developer for your company. You are developing a Web service that will allow customers to access confidential information about their accounts. Client applications that use the Web service will provide an identifier and a password. The Web service will verify credentials by using a Microsoft SQL Server database. You must minimize the risk that account information can be intercepted during transmission. You need to choose an authentication method. What should you do?

A. Use Web Service Enhancements for Microsoft .NET (WSE) to implement the WS-Security specification.

B. Require client applications to encrypt an identifier and password in the SOAP header of all requests.

C. Use basic authentication over SSL/TLS.

D. Use Forms authentication over SSL/TLS.

**Answer: A**

**Question: 52**

You are an application developer for Company.com. You are developing a multithreaded application. Some of the application's threads perform maintenance tasks in the application's database. These maintenance tasks are performed by dedicated assemblies, and the assemblies need to run under different security permissions. The other assemblies of the application must not have the different permissions. You need to ensure that the application's threads have the correct security permissions. You want to achieve this goal without negatively affecting response times for the application. What should you do?

- A. Configure the application to impersonate a user account that has the permissions required by the maintenance assemblies.
- B. Configure code access security policies so that the application has the permissions required by the maintenance assemblies.
- C. Create a separate application domain for the maintenance assemblies.
- D. Start the maintenance assemblies as separate processes.

**Answer: C**

**Explanation**

The logical and physical boundary created around every .NET application by the Common Language Runtime (CLR). The CLR can allow multiple .NET applications to be run in a single process by loading them into separate application domains. The CLR isolates each application domain from all other application domains and prevents the configuration, security, or stability of a running .NET applications from affecting other applications. Objects can only be moved between application domains by the use of remoting.

**Application Domains**

Operating systems and runtime environments typically provide some form of isolation between applications. This isolation is necessary to ensure that code running in one application cannot adversely affect other, unrelated applications. Application domains provide a more secure and versatile unit of processing that the common language runtime can use to provide isolation between applications. Application domains are typically created by runtime hosts, which are responsible for bootstrapping the common language runtime before an application is run.

**Application Domains and Threads**

Application domains form an isolation, unloading, and security boundary for managed code. Threads are the operating system construct used by the common language runtime to execute code. At run time, all managed code is loaded into an application domain and is run by a particular operating system thread. There is not a one-to-one correlation between application domains and threads. Several threads can be executing in a single application domain at any given time and a particular thread is not confined to a single application domain. That is, threads are free to cross application domain boundaries; a new thread is not created for each application domain. At any given time, every thread is executing in one application domain. The run time keeps track of which threads are running in which application domains. You can locate the domain in which a thread is executing at any time by calling the Thread.GetDomain method.

**Question: 53**

You are an application developer for Company.com. You are developing a library assembly that is part of an accounting application. The library includes a serviced component named CompanyProcessing. The application will use COM+ role-based security to restrict access to components. The business rules implemented by the application allow only those users who are members of the COM+ role named Payroll to access the CompanyProcessing component. However, users who are not members of the Payroll role are allowed to access other components in the library. You need to modify your code to enable role verification and ensure that only members of the Payroll role can access the CompanyProcessing component. Which three actions should you perform? (Each correct answer presents part of the solution. Choose three)

- A. Add the following attribute to the library assembly. <Assembly: ApplicationAccessControl(AccessChecksLevel:= \_ AccessChecksLevelOption.ApplicationComponent)>
- B. Add the following attribute to the library assembly. <Assembly: ApplicationAccessControl(AccessChecksLevel:= \_ AccessChecksLevelOption.Application)>

- C. Add the following attribute to the CompanyProcessing component. <ComponentAccessControl(>
- D. Add the following attribute to the CompanyProcessing component. <PrivateComponent()
- E. Add the following attribute to the CompanyProcessing component. <SecurityRole("Payroll")>
- F. Add the following attribute to the CompanyProcessing component.  
<PrincipalPermission(SecurityAction.Demand, Role:="Payroll")>

**Answer: A, C, E**

**Question: 54**

You are an application developer for Company.com. You create an ASP.NET Web application that is hosted on an intranet Web application server named Company1. The application is configured to use Forms authentication. The application requires users to log on by using a name and password. The application stores users names and passwords in a Microsoft SQL Server database that is located on a database server named Company2. The login pages for the application use SSL/TLS encryption. No other pages in the application use SSL/TLS. You need to exam the application to find out if unauthorized users can view user name and password information. What should you do?

- A. Access the application by using a user account that is a member of the SQL Server db\_owner role.
- B. Access the login pages by using HTTP.
- C. Exam the users who enter incorrect user name and password combinations are prevented from accessing the application.
- D. Capture and analyze the network traffic between Company1 and Company2.

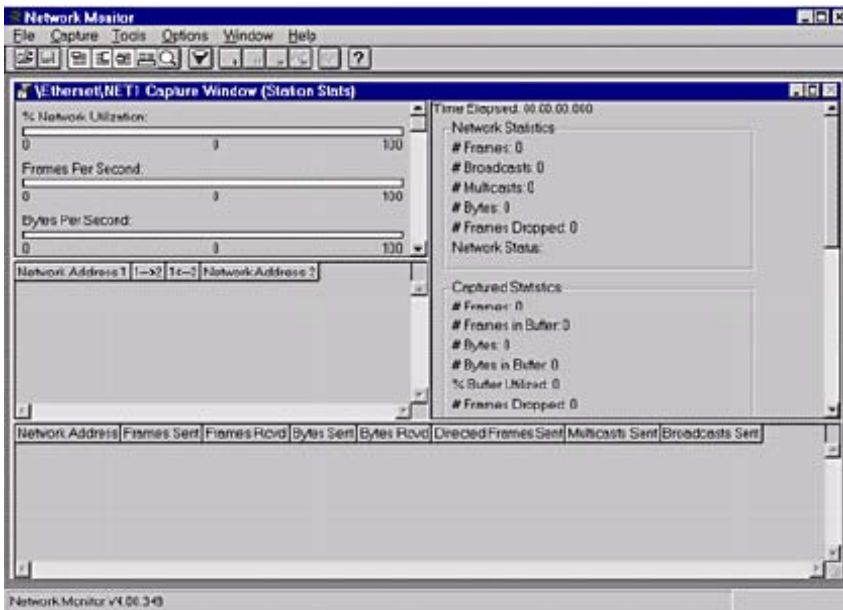
**Answer: D**

**Explanation**

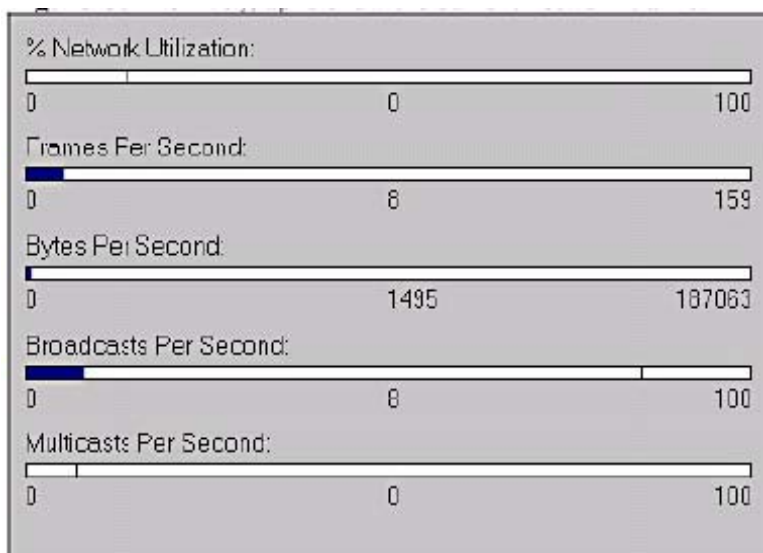
Keeping an eye on the invisible current of network traffic can be a difficult task. One way to do this is by using the Network Monitor tool. This tool allows you to look at the traffic on your network by capturing a sample of the data that's transmitted. Network Monitor also lets you view this sample of network data in a number of useful ways. In this article, we'll show you how to install the Network Monitor and explain how it can help you examine traffic on your network. We'll also take a closer look at the nature of network communications and try to shed some light on the bits of data that flow through all of those wires. Finally, we'll show you how to filter the data you've captured to make it easier to view. nstalling Network Monito y default, Network Monitor isn't lalled when you first install Windows NT. You'll need to install it as a network service before you can use it. To do this, you must access the NT setup CD-ROM. This can either be over the network or in your local CD-ROM drive. To install Network Monitor, launch the Network utility from the Control Panel. Click on the Services tab, and click Add to open the Select Network Service dialog box. Scroll down the list of services until you find the Network Monitor Tools And Agent item. Highlight the entry, and your dialog box should look like the one shown in Figure A. Click OK and type the path to your NT setup files if it's requested. Close the Network utility, and restart your computer when prompted.



Running Network Monitor When you've rebooted your system, NT places the Network Monitor program in the Administrative Tools menu. Launch this program to run it. When you start Network Monitor, it will open the Capture window, as shown in Figure B. **Figure B:** As well as sending out a network alert, Network Monitor logs each occurrence of an alert event.



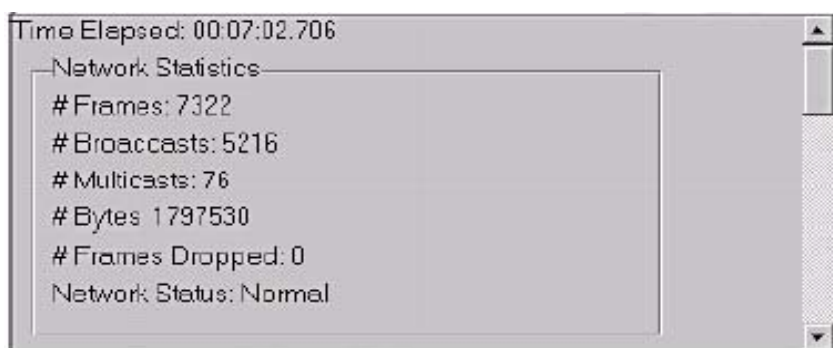
You'll notice that when you start Network Monitor, there's no activity in any of the panes. You must tell Network Monitor when it should start capturing data. We'll explain how to capture data a bit later. First, let's have a closer look at the Capture window. The Capture window consists of four panes. These panes are the Graph pane, the Session Stats pane, the Total Stats pane, and the Station Stats pane. Since the Capture window is the main interface to the Network Monitor, we'll cover the four panes in detail. The Graph pane The Graph pane is located in the upper-left section of the Network Monitor display. This pane displays a graphical representation of the current network activity. The display is real-time and shows three statistics: the percent of network usage, the frames per second, and the bytes per second. When expanded, the Graph pane appears as shown in Figure C.



The % Network Utilization bar graph displays the percentage of your network's available resources that are being used. This bar has a maximum value of 100 percent. A black line appears on the bar to represent the maximum value reached in any capture session. Below network usage lies the Frames Per Second bar graph. This graph displays the number of frames transmitted over your network each second. We'll discuss frames in more detail later. The right-most value of this bar, like those to follow, indicates the maximum value for the current session, unless it's less than 100. The remaining bar graphs--Bytes Per Second, Broadcasts Per Second, and Multicasts Per Second--display the current statistics for the three measures of network traffic. The Session Stats pane When two computers establish communication and send and receive information, it's known as a session. The Session Stats pane displays a summary of the sessions between two systems. The systems are identified by either their names or hardware addresses. This pane contains four columns and is located directly below the Graph pane. The first column lists one of the systems in the session. This system is referred to as 1. The fourth column lists the other system in the session, referred to as 2. The second and third columns list the number of frames that were sent between the two systems. The column labeled 1→2 indicates the frames sent from the system in the first column to the system in the fourth. The column labeled 2→1 displays the frames sent in the other direction. An example of the Session Stats pane is shown in Figure D. **Figure D:** The Session Stats pane displays a summary of the conversations between two systems.

| Network Address 1 | 1→2  | 2→1 | Network Address 2  |
|-------------------|------|-----|--------------------|
| 006008D08F31      | 15   |     | EARTH              |
| 00E01E3E1F32      | 1852 |     | EARTH              |
| EARTH             | 17   |     | *BROADCAST         |
| EARTH             | 5    |     | *NETBIOS Multicast |
| EARTH             | 7    |     | EARTH              |
| Intel BAD29C      | 94   |     | EARTH              |
| JUPITER           | 62   |     | EARTH              |

B. The Total Stats pane The Total Stats pane is located to the right of the Session Stats pane. It contains five groups of network statistics. The first is Network Statistics, which displays information about the total traffic on your network. The next is Captured Statistics, which shows statistics for the frames you've captured. Per Second Statistics calculates the per second averages of the traffic on the network. The last two groups of statistics deal with your system's network card. Network Card (NIC) Statistics lists the supported information about your card. Network Card (NIC) Error Statistics lists any errors that were revealed. At the top of the pane, the total time of the capture is displayed. Figure E shows the Network Statistics group displaying statistics for a capture that lasted a little longer than seven minutes. **Figure E:** The Network Statistics group is just one of five categories of statistics available in the Total Stats



The Station Stats pane On the very bottom of the Capture window, you'll find the Station Stats pane. As shown in Figure F, the Station Stats pane displays network statistics for individual hosts. It also displays the statistics for broadcast and multicast messages received. You can sort columns either descending or ascending by double-clicking on the column head.

| Network Address  | Frames Sent | Frames Recd | Bytes Sent | Bytes Recd | Dropped Frames Sent | Multicasts Sent | Broadcasts Sent |
|------------------|-------------|-------------|------------|------------|---------------------|-----------------|-----------------|
| *BROADCAST       | 0           | 17          | 0          | 1567       | 0                   | 0               | 0               |
| *NETBIOS Multic  | 0           | 5           | 0          | 608        | 0                   | 0               | 0               |
| 00:00:00:08:F3:1 | 15          | 0           | 1053       | 0          | 15                  | 0               | 0               |
| 00:01:02:01:F7:2 | 1062        | 0           | 706326     | 0          | 1062                | 0               | 0               |
| EARTH            | 39          | 2010        | 3135       | 80613      | 7                   | 5               | 17              |
| Intel BAD23C     | 34          | 0           | 1397E      | 0          | 34                  | 0               | 0               |
| JUPITER          | 62          | 0           | 6093       | 0          | 62                  | 0               | 0               |

**Capturing data** When you first launch Network Monitor, the Capture window will be displayed, but the statistics will all be zeroed. In order to capture network data, you must start the capture. To do this, select Capture | Start from the menu bar. Once you've started the capture, the statistics are displayed in real time. While you browse the network or the Internet, you can watch as the statistics begin to accumulate. When you've captured a representative amount of data, select Capture | Stop to halt the capture and prepare the data for analysis. Before you view the captured data, try to resolve all network card numbers to machine names. To do this, select Capture | Find All Names after you've stopped your capture. **Viewing captured data** When monitoring your network traffic, what you're really capturing are the individual units of network communication, or frames. A frame is one part of the message being sent from one computer to another. Most network cards can only handle between 1,500 and 4,000 bytes of information at one time. Because of this, long messages must be broken into a number of smaller chunks. These chunks are then packaged in frames and sent to their destination where they're put back together to form the entire message. Beyond just containing a part of the message, each frame must also have the address of the source and destination computers, because each frame is transmitted individually over the network. Sending a large message using frames is much like sending a car from New York to California using containers that are all the same size. Before the message goes out, you must break it down, put the parts into packages that are separately addressed, including the ship to and return addresses. Once they're received, the containers are opened, accounted for, and reassembled into the original message. When you view the captured data in Network Monitor, you're really seeing the individual frames. Unlike a postal letter, each frame has two destination addresses--a software address and a hardware address. The software address is the arbitrary address assigned to the machine. Depending on the protocol, this address can have different forms. In the TCP/IP protocol, it's the familiar dotted quad IP address. The hardware address is the network adapter fixed address. This address is usually burned onto the adapter by the manufacturer and can't be changed. You may also see the hardware address called a Media Access Control (MAC) address. A system can send two types of frames. The first is a directed transmission. This is a frame that has a specific destination address, like the familiar business envelopes. A directed transmission happens when you're communicating with a specific computer, such as when you request a Web page from a server. The second type of frame is the broadcast transmission. A broadcast transmission is sent to all systems on your network. To do this, broadcast transmissions use a special hardware address. Broadcast transmissions are used when the address for a specific host is unknown, or when a service is available on the network, but the host offering this service may change. Using the Computer dialog box (by choosing Find | Computer) is an example of a broadcast message. Searching for a particular computer name sends a broadcast message to all computers. If a computer's name matches the name in the broadcast transmission, the system sends a directed transmission to the source destination listed in the broadcast frame. **Viewing the frames** To view the details of a captured data set, you select Capture | Display Captured Data from the menu bar, or press [F12]. This displays the Capture (Summary) window, as shown in Figure G. In this window, you see each captured frame. Each line lists the frame's number, time of capture, source address, destination address, protocol, and description. **Figure G:** The Capture (Summary) window displays the captured frames.

| Frame | Time   | Src MAC Addr | Dest MAC Addr | Protocol | Description                          |
|-------|--------|--------------|---------------|----------|--------------------------------------|
| 1     | 4.124  | EARTH        | *BROADCAST    | ARP      | Find Name EARTH                      |
| 2     | 4.124  | EARTH        | EARTH         | ARP      | Name Recognized EARTH                |
| 3     | 5.593  | JUPITER      | EARTH         | ARP      | Session Data, Keep Alive             |
| 4     | 11.124 | 00E011381F32 | EARTH         | SMB      | R NT create & X, FID = 0x0002        |
| 5     | 11.133 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 6     | 11.144 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 7     | 11.152 | 00E011381F32 | EARTH         | SMB      | R close file                         |
| 8     | 11.166 | 00E011381F32 | EARTH         | SMB      | R NT create & X, FID = 0x0001        |
| 9     | 11.179 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 10    | 11.180 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 11    | 11.187 | 00E011381F32 | EARTH         | SMB      | R close file                         |
| 12    | 11.967 | JUPITER      | EARTH         | ARP      | Session Data, Ack, Recv Seq 0x04, 8x |
| 13    | 12.383 | 00E011381F32 | EARTH         | SMB      | R NT create & X, FID = 0x780c        |
| 14    | 12.390 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 15    | 12.502 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 16    | 12.509 | 00E011381F32 | EARTH         | SMB      | R close file                         |
| 17    | 12.416 | 00E011381F32 | EARTH         | SMB      | R NT create & X - NT error: System   |
| 18    | 12.435 | 00E011381F32 | EARTH         | SMB      | R transact2 (Unknown) (response)     |
| 19    | 12.544 | 00E011381F32 | EARTH         | SMB      | R NT create & X, FID = 0x780b        |
| 20    | 12.551 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 21    | 12.562 | 00E011381F32 | EARTH         | SMB      | R transact                           |
| 22    | 12.567 | 00E011381F32 | EARTH         | SMB      | R close file                         |

You can view the contents of an individual frame by double-clicking on any frame in the window. The window will then divide into three frames, as seen in Figure H. The top pane remains the Summary pane. Beneath that, the Detail frame displays the information added to the frame by the various protocols. The bottom pane in the window is the Hex pane. This pane displays the contents of the frame in both ASCII and hexadecimal. **Figure H:** When you double-click on a frame, the Capture window divides into the Summary, Detail, and Hex frames.

| Frame | Time   | Src MAC Addr | Dest MAC Addr | Protocol | Description                          |
|-------|--------|--------------|---------------|----------|--------------------------------------|
| 142   | 65.326 | EARTH        | *BROADCAST    | ARP      | Find Name EARTH                      |
| 143   | 66.077 | WAP5         | EARTH         | LLC      | ?? DSAP=0xF0 SSAP=0xF1 D H(S) = 0x00 |
| 144   | 68.268 | 00E011381F32 | EARTH         | TCP      | A..S., len: 4, seq: 768979921-76     |
| 145   | 69.402 | 00E011381F32 | EARTH         | TCP      | A..S., len: 322, seq: 768979922-76   |
| 146   | 68.527 | 00E011381F32 | EARTH         | TCP      | A..S., len: 4, seq: 528128000-52     |
| 147   | 68.622 | 00E011381F32 | EARTH         | TCP      | A..S., len: 182, seq: 528128001-52   |
| 148   | 68.629 | 00E011381F32 | EARTH         | TCP      | A..S., len: 536, seq: 528128103-52   |

Base frame properties:

ETHERNET: IIType = 0x0000 : Protocol = IP: DOB Internet Protocol

IP: ID = 0x1909, Proto = TCP, Len: 44

TCP: A..S., len: 4, seq: 768979921-768979924, ack: 7491677, win: 8760, src: 90, dest: 12

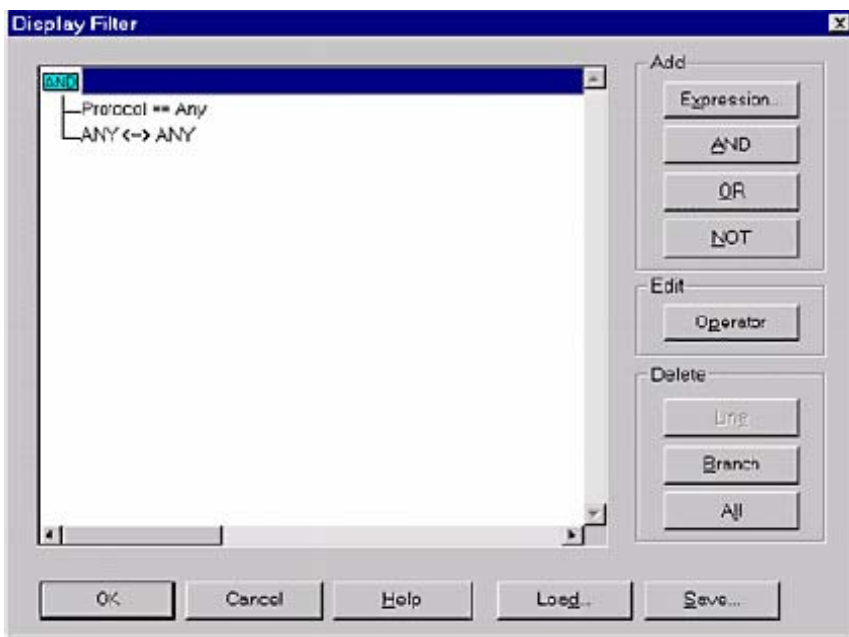
Hex:

```

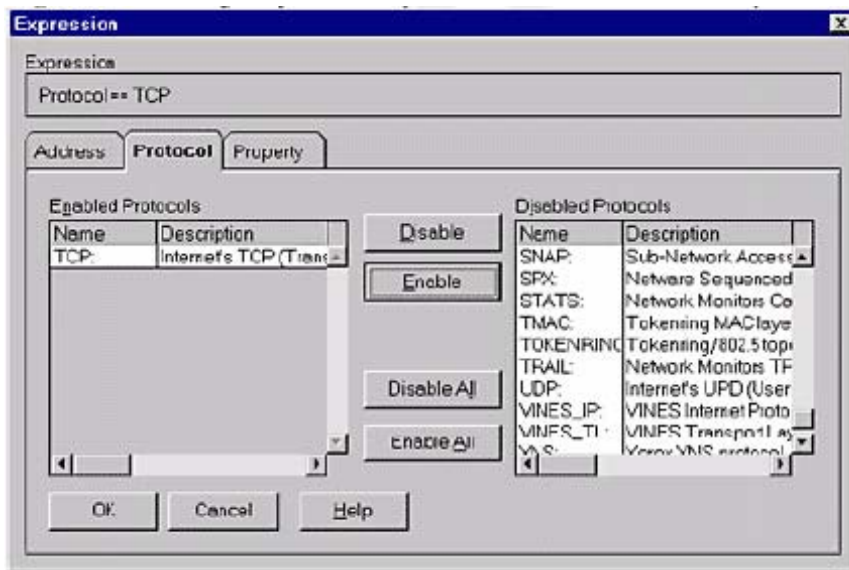
00000000 00 20 EF 29 FD CF 00 10 1E 1F 22 02 00 65 06 .C 1E..a7b02..6
00000010 00 2C E7 D3 40 00 31 06 1C 53 CF 44 87 C2 9B 28 .t181..c-D1-ed
00000020 4B 56 00 50 05 72 2D 05 B3 D1 64 77 A7 15 60 12 [B] 7..T-!..w..
00000030 22 38 89 5F 00 00 02 04 10 D8 06 EF "8_....+..h

```

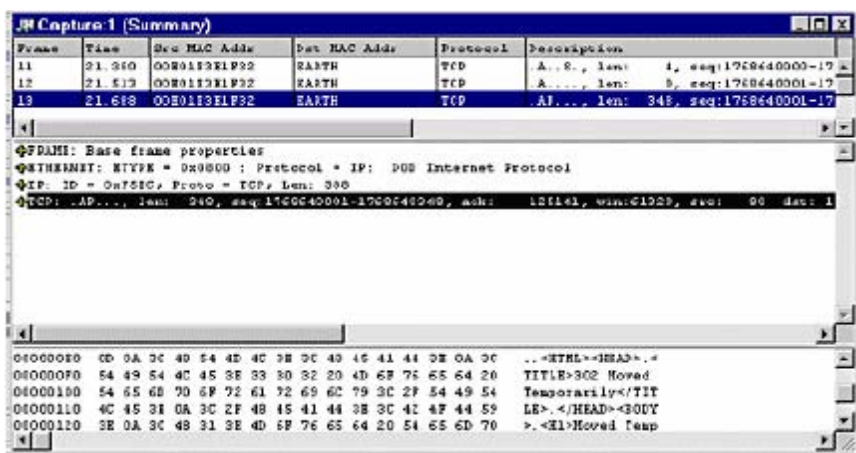
It's beyond the scope of this article to try to explain the possible contents of any individual frame. However, you can find a number of references in the Network Monitor Help pages. Filtering captured data One of the problems with using Network Monitor is that it will capture all of the frames it sees. This can make viewing the individual frames more difficult as they get lost in the abundance of data. One way to limit the types of frames to view is by using Filters. You can filter captured data by filtering the frames according to the protocol used to transmit them. Another way is to display only data transmitted from a specific computer or group of computers. We'll show you how to configure a filter to display only TCP/IP frames. To display only the frames transmitted by a specific computer or protocol, you must first capture some data. As described above, you do this by starting Network Monitor and choosing Capture | Start from the menu bar. After you've begun the capture, surf the 'Net using a Web browser to generate some network traffic. After a minute or so, you should have enough data. Select Capture | Stop from the menu bar to end the capture. Next, display the Capture Summary window by choosing Capture | Display Captured Data from the menu bar. Select Display | Filter to launch the Display Filter dialog box, as shown in Figure I. By default, the Filter is configured to display frames of any protocol. **Figure I:** The Display Filter dialog box is configured by default to display frames from any machine sent with any protocol.



Double-click on Protocol == Any to launch the Expression dialog box. Click Disable All to remove all protocols from the filter. Then, scroll down the Disabled Protocols list until you find the TCP protocol. Highlight this protocol by clicking on it with your mouse, and then click Enable. Your dialog box should look like the one shown in Figure J. Click OK to apply the expression and close the dialog box. **Figure J:** Enabling only the TCP protocol allows us to view only those frames transmitted using that protocol.



The Display Filter dialog box now shows that you'll be displaying only those frames that were prepared using the TCP protocol. Click OK to close this dialog box, and return to the Capture window summary display. In the summary window, only frames sent with the TCP protocol are displayed. Select one of the frames and double-click on it. View the contents of the frame in the Hex pane. Use your arrow keys to scroll through the frames while watching the Hex pane. You should see some recognizable data appear in the pane. In Figure K, you can see that the frame selected contained HTML code, which is probably part of the transmission of a Web page. **Figure K:** This frame contains HTML code and it was probably just one frame of a transmission of a Web page.



Conclusion Network Monitor is a powerful tool to use when analyzing network traffic. It can give you an overview of the performance of your system, as well as allow you to view individual frames. Using the techniques described here, you can peek at the hidden stream of data that runs inside your network cables.

**Question: 55**

You are an application developer for Company.com. You are developing an application that will be distributed to partner companies that do business with Company. The partner companies must be able to verify the authenticity of the application's assemblies before they will install the application.

You need to ensure that the assemblies meet the requirements of the partner companies, You want your solution to minimize the number of additional configuration steps required by the partner companies.

What should you do?

- A. Use a certificate issued by Company's internal, self-signed certification authority (CA) to digitally sign the assemblies.
- B. Use a certificate issued by a third-party commercial certification authority (CA) to digitally sign the assemblies.
- C. Run the PEVerify tool before distributing the application to partner companies.
- D. Run the Software Publisher Certificate Exam tool before distributing the application to partner companies.

**Answer: B**

## Explanation Choosing Security Solutions That Use Public Key Technology

Software that is downloaded from the Internet to users' computers can contain programs such as viruses and Trojan horses that are designed to cause malicious damage or provide clandestine network access to intruders. As networks become more interconnected, malicious software and viruses also become a threat to intranets. To help counter this growing threat, you can digitally sign the software that you distribute on your intranets or the Internet to ensure its integrity and to assure others that the software can be trusted. Signed software ensures that users can verify the origin of the software, as well as verify that no one has tampered with it. Microsoft developed the Microsoft® Authenticode® technology, which enables developers to digitally sign software. The last thing developers do before they release software is digitally sign the software. Any modification to the software after it is signed invalidates the digital signature. By using Authenticode technology, code signers who own valid X.509 version 3 code-signing certificates can sign software with their private key. Several other third-party code signing technologies also use digital certificates to enable code signing. Code Signing Within Your Organization Executable programs, scripts, and ActiveX® controls that are distributed in Windows 2000 domains should be digitally signed by trusted developers. To protect your network from malicious programs and viruses, you can configure Internet Explorer to specify security settings for the Internet, local intranet, Trusted sites, and Restricted sites security zones. You can specify security settings that prevent users from downloading and running unsigned software from any security zone. You can also configure Internet Explorer to trust specific software publishers so that any software that is signed by these publishers is downloaded automatically without notifying the users. For more information about Internet

Explorer security, see the *Microsoft® Windows® 2000 Server Resource Kit Internet Explorer Resource Guide*. In addition, you can configure Public Key Group Policy to specify the CAs for code signing that are trusted in your organization. You can trust software publishing certificates that are issued either by commercial CAs or by your CAs. You can also create and use CTLs to establish trust in the domain for code-signing certificates. You can use Certificate Services to issue code-signing certificates to the developers who sign software for distribution on your intranet. Code Signing on the Internet When software is distributed over the Internet, users are more likely to trust software that is signed by a publisher whose code-signing certificates ("software publisher certificates") have been issued by a reputable commercial CA. Using commercial CAs also removes the liability placed on your organization when you assume the responsibilities of a commercial CA for external software distribution. Therefore, if you distribute software on the Internet, consider obtaining the services of a commercial CA to issue digital signing certificates to your external software developers. Consider providing special protection for the private keys that are used to sign code. If someone obtains access to a private key for code signing, they can impersonate your organization, distribute signed but defective or malicious code, and damage your organization's reputation. Some third-party vendors offer smart card solutions that enable code signing with smart cards. You can establish a smart card program for code signers that provides additional protection for their private keys. Automating Code Signing and Software Distribution You can build custom applications to automate code signing and the distribution of software within your organization or to external Web sites. Internal and external developers or program managers who have valid code-signing certificates can use custom applications to submit code to be signed automatically and processed for distribution. Deploying code-signing applications includes the following activities:

- Installing CAs that issue code-signing certificates or obtaining certificate services from a commercial CA.

- Developing the custom applications for code signing and software distribution.

- Issuing code-signing certificates to the appropriate developers or program managers.

- Configuring the software distribution infrastructure and services. For example, you might use Active Server Pages (ASP) technology and Internet Information Services to build code-signing and software distribution Web sites. You might configure one-to-one certificate mapping to grant permission for use of the Web site to users who have valid code-signing certificates. Users who do not have valid code-signing certificates are not permitted to use the site to submit code for signing and distribution.

#### Question: 56

You are an application developer for Company.com. You are developing an ASP.NET Web application on your Microsoft Windows XP Professional client computer. Your computer has IIS

5.1 installed and is hosting the development project. Your computer also has the most recent version of the Microsoft .NET Framework installed. The completed application will run under IIS

2

You need to ensure that your examsing activities accurately reflect the security configuration of the production environment.

What should you do before examsing the application?

- A. Install the .NET Framework 1.0 on the production Web server.
- B. Deploy the application to a Microsoft Windows Server 2003 exam computer.
- C. Install the most recent security updates for Microsoft Visual Studio .NET 2003 on your client computer.
- D. Disable the ASPNET user account on your client computer.

**Answer: B**

#### Explanation

It is an industry best practice to never place unexamined code in a live environment. The best practice is to setup a lab environment that mimics the production and perform examsing there prior to deployment. It is easier to rollback a exams environment than the production environment. In most cases a rollback to a pristine exams environment is mandatory to ensure the subsequent examsing is not impacted by previous exams with different applications. Since IIS6 can be configured to be much more restrictive than earlier versions, this validation is required. Software Examsing Best Practices <http://www.chillarege.com/authwork/papers1990s/ExamsingBestPractice.pdf> Desktop Deployment Web Seminar <http://www.microsoft.com/downloads/details.aspx?FamilyID=2bc24b8c-ba04-45c3-bcd012af0e718e79&DisplayLang=en>

#### Question: 57

You are an application developer for Company.com. You are examining an application that was developed by another developer.

The application maintains its own list of authorized users. Each user is assigned a security level of 1, 2, or 3. When a new user account is created, the security level for that user is entered into a text box. The new user account information is saved in a Microsoft SQL Server table by using a stored procedure. You verify that user accounts that have any of the three security levels can perform only the intended actions within the application.

You need to identify any security vulnerabilities in the portion of the application that creates new user accounts.

What should you do first?

- A. Use SQL Query Analyzer to create a new user account that has a security level of **2**. Exam the application to see if the new user account can log on to the application.
- B. Create a new user account that has a security level other than **1, 2, or 3**. Exam the application to see what the new user account can do.
- C. Use Osq.exe to call the stored procedure and create a new user account that has a security level of **3**. Exam the application to see what the new user account can do.
- D. Create a new user account that has a security level of **3**. Exam the application to see what the new user account can do.

**Answer: B Explanation**

Security examining is about validating your application's security services and identifying potential security flaws. This section contains important examining recommendations for verifying that you have created a securable application. Since attackers have no standard method of breaking into things, there are no standard methods of conducting security examining. Also, there are few tools available at this time to examine security aspects thoroughly. Since a functional bug in an application can also represent a potential security flaw, you need to conduct functional examining prior to conducting security examining. It is important to note that security examining will not prove conclusively that an application is secure. Instead, it serves only to validate the effectiveness of instituted countermeasures, which were chosen based upon presumptions that were made during the threat analysis phase. Provided below are some suggestions for examining the securability of your application. There are some security issues you should be aware of when you examine your smart documents. These security measures, described in the Security section, are in place to provide security for Microsoft® Office 2003 users. However, during examining, you may want to disable the XML expansion pack security check, if possible, or you may want to create an examining environment that meets the security requirements of your users. The following topics provide additional information about security within a development and examining environment: Disabling the XML Expansion Pack Security Check Digital Code Signing for Examining Purposes Creating a Digital Certificate for Examining Purposes Delay Signing a Smart Document Assembly Examining a Signed XML Expansion Pack

### **Exams for Buffer Overflows**

One of the first security bugs exploited in computer history was a buffer overflow. Buffer overflows continue to be one of the most dangerous and most commonly occurring weaknesses. Attempts to exploit this type of vulnerability can result in problems ranging from crashing the application to an attacker inserting and executing malignant code in the application process. When writing data to buffers, it is imperative that developers not write more to the buffer than it can possibly hold. If the amount of data being written exceeds the buffer space that has been allocated, a buffer overflow occurs. When a buffer overflow occurs, data is written into parts of memory that may be allocated for other purposes. A worst-case scenario is when the buffer overflow contains malicious code that is then executed. Buffer overflows account for a large percentage of security vulnerabilities.

### **Conduct source code security reviews**

Depending upon the sensitivity of the application in question, it might be prudent to conduct a security audit of the application source code. A source code audit should not be confused with a code review. The purpose of a standard code review is to identify general code defects that affect the functionality of the code. The purpose of a source code security review is to identify security flaws, intentional or otherwise. Such a review would be especially warranted when developing applications that handle financial transactions or provide for public safety.

### **Validate contingency plans**

There will always be a potential that an application's security defenses can be breached and it is only prudent that contingency plans

are in place and validated. What steps will be taken if a virus is detected on your application server or in your data center? When security is thwarted, reactions must occur rapidly to prevent further damage. Find out if your contingency plans will work before they must be battle-examined.

### **Attack your application**

Examsmen are accustomed to tormenting applications in an attempt to make them fail. Hacking your own application is a similar, but more focused, process. When attempting to attack your application, you should be looking for exploitable flaws that represent a weak spot in your application's defenses.

### **Question: 58**

You are an application developer for Company.com. Part of the application that you are developing accepts user input from a TextBox control. The information entered by the user must be alphanumeric only, and it must contain no symbols or punctuation.

You need ensure that the user's input contains only the appropriate data before using the input elsewhere in the application. Your solution must not require users of the application to take additional steps when entering data.

What should you do?

- A. Modify the TextChanged event handler of the TextBox control so that the Text property of the text box is cleared whenever a non-alphanumeric character is detected.
- B. Use the following regular expression to modify the user's input. `[^\w\.\@-]`
- C. Store the user's input in a variable named userInput. Use the following expression to modify the user's input. `userInput.Replace("@-","")`
- D. Convert the user's input to all lowercase characters.

**Answer: B**

### **Explanation**

Never Trust User Input! I know this injunction sounds harsh, as if people are out to get you. But many are. If you accept input from users, either directly or indirectly, it is imperative that you validate the input before using it, because people will try to make your application fail by tweaking the input to represent invalid data. The first golden rule of user input is, *All input is bad until proven otherwise*. Typically, the moment you forget this rule is the moment you are attacked. In this section, we'll focus on the many ways developers read input, how developers use the input, and how attackers try to trip up your application by manipulating the input. Let me introduce you to the second golden rule of user input: *Data must be validated as it crosses the boundary between untrusted and trusted environments*. By definition, trusted data is data you or an entity you explicitly trust has complete control over; untrusted data refers to everything else. In short, any data submitted by a user is initially untrusted data. The reason I bring this up is many developers balk at checking input because they are positive that the data is checked by some other function that eventually calls their application, and they don't want to take the performance hit of validating the data. But what happens if the input comes from a source that is not checked, or the code you depend on is changed because it assumes some other code performs a validity check?

### **NOTE**

A somewhat related question is, what happens if an honest user simply makes an input mistake that causes your application to fail? Keep this in mind when I discuss some potential vulnerabilities and exploits. I once reviewed a security product that had a security flaw because a small chance existed that invalid user input would cause a buffer overrun and stop the product's Web service. The development team claimed that it could not check all the input because of potential performance problems. On closer examination, I found that not only was the application a critical network component—and hence the potential damage from an exploit was immense—but also it performed many time-intensive and CPU-intensive operations, including public-key encryption, heavy disk I/O, and authentication. I doubted much that a half dozen lines of input-checking code would lead to a performance problem. As it turned out, the code did indeed cause no performance problems, and the code was rectified. *User Input Remedies* As with all user input issues, the first rule is to determine which input is valid and to reject all other input. (Have I said that enough times?) Other not-so-paranoid options exist and offer more functionality with potentially less security. I'll discuss some of these also.

**A Simple and Safe Approach: Be Hardcore About Valid Input**

In the cases of the Web-based form and SQL examples earlier, the valid characters for a username can be easily restricted to a small set of valid characters, such as A-Za-z0-9. The following server-side JScript snippet shows how to construct and use a regular expression to parse the username at the server: // Determine whether username is valid. // Valid format is 1 to 32 alphanumeric characters. var reg = /^[A-Za-z0-9]{1,32}\$/g; if (reg.exams(Request.form("name")) > 0) { // Cool! Username is valid. } else { // Not cool! Username is invalid. } A Regular Expression Rosetta Stone Regular expressions are incredibly powerful, and their usefulness extends beyond just restricting input. They constitute a technology worth understanding for solving many complex data manipulation problems. I write many applications, mostly in Perl and C#, that use regular expressions to analyze log files for attack signatures and to analyze source code for security defects. *Regular Expressions in Managed Code* Most if not all applications written in C#, Managed C++, Microsoft Visual Basic .NET, ASP.NET, and so on have access to the .NET Framework and as such can use the *System.Text.RegularExpressions* namespace. I've already outlined its syntax earlier in this chapter. However, for completeness, following are C#, Visual Basic .NET, and Managed C++ examples of the date extraction code I showed earlier in Perl.

**C# Example**

```
// C# Example String s = @"We leave at 12:15pm for Mount Doom."; Regex r = new
Regex(@".*(\d{2}:\d{2}[ap]m)", RegexOptions.IgnoreCase); if (r.Match(s).Success) Console.WriteLine(r.Match(s).Result("$1"));
```

**Visual Basic .NET Example**

```
' Visual Basic .NET example Imports System.Text.RegularExpressions . Dim s As String Dim r As Regex s = "We leave at 12:15pm
for Mount Doom." r = New Regex(".*(\d{2}:\d{2}[ap]m)", RegexOptions.IgnoreCase) If r.Match(s).Success Then
Console.WriteLine(r.Match(s).Result("$1")) End If
```

**Managed C++ Example**

```
// Managed C++ version #using <mcorlib.dll> #include <tchar.h> #using <system.dll> using namespace System; using namespace
System::Text; using namespace System::Text::RegularExpressions; . String *s = S"We leave at 12:15pm for Mount Doom."; Regex *r
= new Regex(".*(\d{2}:\d{2}[ap]m)", IgnoreCase); if (r->Match(s)->Success) Console::WriteLine(r->Match(s)->Result(S"$1")); Note
that the same code applies to ASP.NET because ASP.NET is language-neutral.
```

*Regular Expressions in Script*

The base JavaScript 1.2 language supports regular expressions by using syntax similar to Perl. Netscape Navigator 4 and later and Microsoft Internet Explorer 4 and later also support regular expressions. var r = /.\*(\d{2}:\d{2}[ap]m)/; var s = "We leave at 12:15pm for Mount Doom."; if (s.match(r)) alert(RegExp.\$1); Regular expressions are also available to developers in Microsoft Visual Basic Scripting Edition (VBScript) version 5 via the *RegExp* object: Set r = new RegExp r.Pattern = ".\*(\d{2}:\d{2}[ap]m)" r.IgnoreCase = True Set m = r.Execute("We leave at 12:15pm for Mount Doom.") MsgBox m(0).SubMatches(0) \w ----> matches any word character, equivalent to [a-zA-Z0-9]

**Regular Expressions**

Regular expressions are a concise and flexible notation for finding and replacing patterns of text. The regular expressions used within Visual Studio are a superset of the expressions used in Visual C++ 6.0, with a implied syntax. You can use the following regular expressions in the Find, Replace, Find in Files or Replace in Files dialog boxes to refine and expand your search. **Note** You must select the Use check box in the Find, Replace, Find in Files, and Replace in Files dialog boxes before using any of the following expressions as part of your search criteria. The following expressions can be used to match characters or digits in your search string:

| Expression             | Syntax | Description                                                   |
|------------------------|--------|---------------------------------------------------------------|
| Any character          | .      | Matches any one character except a line break.                |
| Maximal - zero or more | *      | Matches zero or more occurrences of the preceding expression. |
| Maximal - one or more  | +      | Matches at least one occurrence of the preceding expression.  |

|                        |   |                                                                                                       |
|------------------------|---|-------------------------------------------------------------------------------------------------------|
| Minimal - zero or more | @ | Matches zero or more occurrences of the preceding expression, matching as few characters as possible. |
| Minimal - one or more  | # | Matches one or more occurrences of the preceding expression, matching as few characters as possible.  |

|                      |        |                                                                                                                                                                 |
|----------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Repeat n times       | ^n     | Matches n occurrences of the preceding expression. For example, [0-9]^4 matches any 4-digit sequence.                                                           |
| Set of characters    | []     | Matches any one of the characters within the []. To specify a range of characters, list the starting and ending character separated by a dash (-), as in [a-z]. |
| Character not in set | [^...] | Matches any character not in the set of characters following the ^.                                                                                             |
| Beginning of line    | ^      | Anchors the match to the beginning of a line.                                                                                                                   |
| End of line          | \$     | Anchors the match to the end of a line.                                                                                                                         |
| Beginning of word    | <      | Matches only when a word begins at this point in the text.                                                                                                      |
| End of word          | >      | Matches only when a word ends at this point in the text.                                                                                                        |
| Grouping             | ()     | Groups a subexpression.                                                                                                                                         |
| Or                   |        | Matches the expression before or after the OR symbol ( ). Mostly used within a group.                                                                           |

The following table lists the syntax for matching by standard Unicode character properties. The two-letter abbreviation is the same as listed in the Unicode character properties database. These may be specified as part of a character set. For example, the expression [:Nd:Nl:No] matches any kind of digit.

| Expression       | Syntax | Description                                                                        |
|------------------|--------|------------------------------------------------------------------------------------|
| Uppercase letter | :Lu    | Matches any one capital letter. For example, :Luhe matches "The" but not "the".    |
| Lowercase letter | :Ll    | Matches any one lower case letter. For example, :Llhe matches "the" but not "The". |

|                           |     |                                                                                                                                          |
|---------------------------|-----|------------------------------------------------------------------------------------------------------------------------------------------|
| Title case letter         | :Lt | Matches characters that combine an uppercase letter with a lowercase letter, such as Nj and Dz.                                          |
| Modifier letter           | :Lm | Matches letters or punctuation, such as commas, cross accents, and double prime, used to indicate modifications to the preceding letter. |
| Other letter              | :Lo | Matches other letters, such as gothic letter ahsa.                                                                                       |
| Decimal digit             | :Nd | Matches decimal digits such as 0-9 and their full-width equivalents.                                                                     |
| Letter digit              | :NI | Matches letter digits such as roman numerals and ideographic number zero.                                                                |
| Other digit               | :No | Matches other digits such as old italic number one.                                                                                      |
| Open punctuation          | :Ps | Matches opening punctuation such as open brackets and braces.                                                                            |
| Close punctuation         | :Pe | Matches closing punctuation such as closing brackets and braces.                                                                         |
| Initial quote punctuation | :Pi | Matches initial double quotation marks.                                                                                                  |
| Final quote punctuation   | :Pf | Matches single quotation marks and ending double quotation marks.                                                                        |

|                       |     |                                           |
|-----------------------|-----|-------------------------------------------|
| Dash punctuation      | :Pd | Matches the dash mark.                    |
| Connector punctuation | :Pc | Matches the underscore or underline mark. |

In addition to the standard Unicode character properties, the following additional properties may be specified. These properties may be specified as part of a character set.

| Expression            | Syntax | Description                                                                                       |
|-----------------------|--------|---------------------------------------------------------------------------------------------------|
| Alpha                 | :Al    | Matches any one character. For example, :Alhe matches words such as "The", "then", and "reached". |
| Numeric               | :Nu    | Matches any one number or digit.                                                                  |
| Punctuation           | :Pu    | Matches any one punctuation mark, such as ?, @, ', and so on.                                     |
| White space           | :Wh    | Matches all types of white space, including publishing and ideographic spaces.                    |
| Bidi                  | :Bi    | Matches characters from right-to-left scripts such as Arabic and Hebrew.                          |
| Hangul                | :Ha    | Matches Korean Hangul and combining Jamos.                                                        |
| Hiragana              | :Hi    | Matches hiragana characters.                                                                      |
| Katakana              | :Ka    | Matches katakana characters.                                                                      |
| Ideographic/Han/Kanji | :Id    | Matches ideographic characters, such as Han and Kanji.                                            |

.NET Framework Regular Expressions Provides a brief introduction to .NET regular expressions. Regular Expressions as a Language Provides an overview of the programming-language aspect of regular expressions. Regular Expression Classes Provides detailed information and code examples illustrating how to use the regular expression classes. Details of Regular Expression Behavior Provides detailed information about the capabilities and behavior of .NET Framework regular expressions. Regular Expression Examples Provides code examples illustrating typical uses of regular expressions. System.Text.RegularExpressions Provides class-library reference information for the .NET Framework System.Text.RegularExpressions namespace. Regular Expression Validator Control Sample The regular expression validator control shown here extends the base validator control described in the Base Validator Control Sample. This validator adds the following functionality to the base validator:

It exposes a property named ValidationExpression that allows a user (page developer) to specify a

regular expression.

It overrides the `EvaluateIsValid` method (defined as an abstract method in `BaseDomainValidator`) to provide logic to determine whether the field to validate matches the pattern specified by the regular expression.

It overrides `AddAttributeErrorMessageRender` (inherited from `WebControl`) to provide a client-side handler for the evaluation logic. The client-side handler is a function defined in the script library.

### Question: 59

You are an application developer for Company.com. You are developing a method for a Windows Forms application. The method will be used to access local files. The files are located in a folder and its subfolders on drive C of the client computer. All the computers in Company use the NTFS file system exclusively.

The design document for the application specifies the path location for the folder that contains the files. The design document also specifies that the path to the files will be provided as an input parameter to the method. The parameter will be a string parameter. You need to prevent users of the application from accessing any files that are not contained in the specified folder.

Which two actions should you include in the method? (Each correct answer presents part of the solution. Choose two)

- A. Reject any path string that includes a tilde (~).
- B. Reject any path string that includes a folder location that does not match the specified folder location.
- C. Reject any path string that includes either a colon (:) or a backslash (\).
- D. Reject any path string that includes either a `\\.\\` sequence or a `\\` sequence.

**Answer: B, D**

### Explanation

Directory traversal can be accomplished using the '~' and the '\\.', the '\\' can be used to create a connection to any available sharename. Checking for the valid path string can also be accomplished using regular expressions and simply using the pathname. Canonicalization checks must also be made. BOLD indicates what the command resolved to: `C:\>cd \\.windows C:\WINDOWS>cd progra~1` The system cannot find the path specified. `C:\WINDOWS>cd \\.progra~1 C:\PROGRA~1>cd \\windows '\\windows' CMD does not support UNC paths as current directories. C:\PROGRA~1>cd \\.\\.system32` The system cannot find the path specified. `C:\PROGRA~1>cd \\.\\.system32` The system cannot find the path specified. `C:\PROGRA~1>cd \\.\\.windows\system32 C:\WINDOWS\system32>cd \\ '\\'` CMD does not support UNC paths as current directories. `C:\WINDOWS\system32>cd \\.\\.\\wutemp '\\.\\.\\wutemp' CMD does not support UNC paths as current directories. C:\WINDOWS\system32> cd progra~1 : \\.\\micros~1 C:\WINDOWS\system32\MICROS~1> cd \\.progra~1 : \\.\\micros~1` The system cannot find the path specified. `C:\WINDOWS\system32>cd \\. C:\>cd \\.\\.windows\system32 C:\WINDOWS\system32>cd C:\>cd \\.wind~1\system32` The system cannot find the path specified. `C:\>cd windows:system32` The system cannot find the path specified.

### Directory Traversal

HTTP exploits use the Web server software to perform malicious activities. Directory traversal is one such exploit which lets attackers access restricted directories, execute commands and view data outside the normal Web server directory where the application content is stored.

### Detailed description

Attackers use directory traversal attacks to try to access restricted Web server files residing outside of the Web server's root directory. The basic role of Web servers is to serve files. Files can be static, such as image and HTML files, or dynamic, such as ASP and JSP files. When the browser requests a dynamic file, the Web server first executes the file and then returns the result to the browser. Hence, dynamic files are actually files executed on the Web server. To prevent users from accessing unauthorized files on the Web server, Web servers provide two main security mechanisms: the root directory and access controls lists. The root directory

limits users' access to a specific directory in the Web server's file system. All files placed in the root directory and in its sub-directories are accessible to users. To limit users' access to specific files within the root directory, administrators use access control lists. Using access control lists, administrators can determine whether a file can be viewed or executed by users, as well as other access rights. The root directory prevents attackers from executing files such as cmd.exe on Windows platforms or accessing sensitive files such as the "passwd" password file on Unix platforms, as these files reside outside of the root directory. The Web server is responsible for enforcing the root directory restriction. By exploiting directory traversal vulnerabilities, attackers step out of the root directory and access files in other directories. As a result, attackers might view restricted files or execute powerful commands on the Web server, leading to a full compromise of the Web server. A directory traversal vulnerability can exist either in the commercial Web server itself or in the Web application code executed on the Web server. In the case of Web application code, dynamic pages usually receive input from browsers. Here is an example of such an HTTP request: <http://www.acme-hackme.com/online/getnews.asp?item=20March2003.html> In this example, the dynamic page requested by the browser is called getnews.asp and the browser sends the Web server the parameter item with a value of 20March2003.html. When executed by the Web server, getnews.asp retrieves the file 20March2003.html from the Web server's file system, renders it and sends it back to the browser which presents it to the user. A skilled attacker will immediately identify the potential problem in this request as the value of the parameter ends with a file extension, in this case ".html". The attacker will then assume that the dynamic page retrieves the file from the file system and uses it. By ending the following URL to the Web server: <http://www.acme-hackme.com/online/getnews.asp?item=../../../../WINNT/win.ini> the attacker causes getnews.asp to retrieve the file ../../../../WINNT/win.ini from the file system and send it to the attacker's browser. The term "../" stands for "one directory up". This is a common operating system directive. Therefore, the string ../../../../WINNT/win.ini means "go four directories up and retrieve the file win.ini from there". The attacker needs to guess how many directories to climb in order to get to the desired directory. (In this example the attacker tries to get to "C:\") and is assuming that the Web server's root directory is located four directories below "C:\"). Guessing the exact combination is very easy. The attacker simply sends multiple requests until the desired result is achieved. The directory traversal vulnerability occurs when programmers fail to validate input received from browsers. In the above example, the getnews.asp code does not validate that the value of the item parameter does not exceed from the root directory. The directory traversal vulnerability actually bypasses the Web server's root directory restriction by introducing bad code into the Web server. Web applications are not the only source of directory traversal vulnerabilities in your Web site. Some vulnerabilities exist within the Web server. These vulnerabilities can be part of sample files (e.g., sample ASP files) that exist on the Web server, or can be incorporated into the Web server software. For example, some earlier versions of the Microsoft IIS Web server included directory traversal vulnerabilities that allow attackers to fully compromise the Web server by executing files on the server. For example, the following URL: <http://www.acme-hackme.com/scripts/..%5c../winnt/system32/cmd.exe?/c+dir+c:\> would execute the cmd.exe file (operating system shell) and run the "dir c:\" command which lists all files in the C:\ directory. Notice the string "%5c" that appears in the URL. This is a Web server escape code. Escape codes are used to represent normal characters in the form of %nn, where nn stands for a two-digit number. The escape code "%5c" represents the character "\". The problem is that the IIS root directory enforcer did not check for escape codes and allowed that request to execute. The Web server's operating system understands escape codes and executes the command. Escape codes are also very useful for bypassing poorly written filters enforced on input received from users. If the filter looks for "../", then the attacker could easily change the input to "%2e%2e/". This has the same meaning as "../", but is not detected by the filter. The escape code %2e represents the character "." (dot). What makes Web-based canonicalization issues so prevalent and hard to defend against is the number of ways you can represent any character. For example, any character can be represented in a URL or a Web page by using one or more of the following mechanisms:

- The "normal" 7-bit or 8-bit character representation, also called US-ASCII

- Hexadecimal escape codes

- UTF-8 variable-width encoding

- UCS-2 Unicode encoding

- Double encoding

- HTML escape codes (Web pages, not URLs)

### *7-Bit and 8-Bit ASCII*

I trust you understand the 7-bit and 8-bit ASCII representations, which have been used in computer systems for many years.

### *Hexadecimal Escape Codes*

Hex escapes are a way to represent a possibly nonprintable character by using its hexadecimal equivalent. For example, the space character is %20, and the pounds sterling character (£) is %A3. You can use this mapping in a URL such as <http://www.northwindtraders.com/my%20document.doc>, which will open my document.doc on the Northwind Traders Web site;

<http://www.northwindtraders.com/my%20document%2Edoc> will do likewise. In Chapter 8, I mentioned a canonicalization bug in eEye's SecureIIS tool. The tool looked for certain words in the client request and rejected the request if any of the words were found

in the request. However, an attacker could hex escape any of the characters in the request, and the tool would not reject the requests, essentially bypassing the security mechanisms. *UTF-8 Variable-Width Encoding* Eight-bit Unicode Transformation Format, UTF-8, as defined in RFC 2279 ([www.ietf.org/rfc/rfc2279.txt](http://www.ietf.org/rfc/rfc2279.txt)), is a way to encode characters by using one or more bytes. The variable-byte sizes allow UTF-8 to encode many different byte-size character sets, such as 2-byte Unicode (UCS-2), 4-byte Unicode (UCS-4), and ASCII, to name but a few. However, the fact that one character can potentially map to multiple-byte representations is problematic.

### How UTF-8 Encodes Data

UTF-8 can encode *n*-byte characters into different byte sequences, depending on the value of the original characters. For example, a character in the 7-bit ASCII range 0x00-0x7F encodes to *07654321*, where *0* is the leading bit, set to 0, and *7654321* represents the 7 bits that make up the 7-bit ASCII character. For instance, the letter *H*, which is 0x48 in hex, or 1001000 in binary, becomes the UTF-8 character *01001000*, or 0x48. As you can see, 7-bit ASCII characters are unchanged by UTF-8. Things become a little more complex as you start mapping characters beyond the 7-bit ASCII range, all the way up to the top of the Unicode range, 0x7FFFFFFF. For example, any character in the range 0x80-0x7FF encodes to *110xxxxx 10xxxxxx*, where *110* and *10* are predefined bits and each *x* represents one bit from the character. For example, pounds sterling is 0xA3, which is 10100011 in binary. The UTF-8 representation is *11000101 10000011*, or 0xC5 0x83. However, it doesn't stop there. UTF-8 can encode larger byte-size characters. Table 12-2 outlines the mappings.

Table 12-2 UTF-8 Character Mappings

| CharacterRange         | Encoded Bytes                                          |
|------------------------|--------------------------------------------------------|
| 0x00000000- 0x0000007F | 0xxxxxxx                                               |
| 0x00000080- 0x000007FF | 110xxxxx 10xxxxxx                                      |
| 0x00000800- 0x0000FFFF | 1110xxxx 10xxxxxx 10xxxxxx                             |
| 0x00010000- 0x001FFFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx                    |
| 0x00200000- 0x03FFFFFF | 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx           |
| 0x04000000- 0x7FFFFFFF | 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx, 10xxxxxx |

And this is where the fun starts; it is possible to represent a character by using any of these mappings, even though the UTF-8 specification warns against doing so. All UTF-8 characters should be represented in the shorexams possible format. For example, the only valid UTF-8 representation of the ? character is 0x3F, or 00111111 in binary. On the other hand, an attacker might try using illegal nonshorexams formats, such as these:

0xC0 0xBF  
0xE0 0x80 0xBF  
0xF0 0x80 0x80 0xBF  
0xF8 0x80 0x80 0x80 0xBF

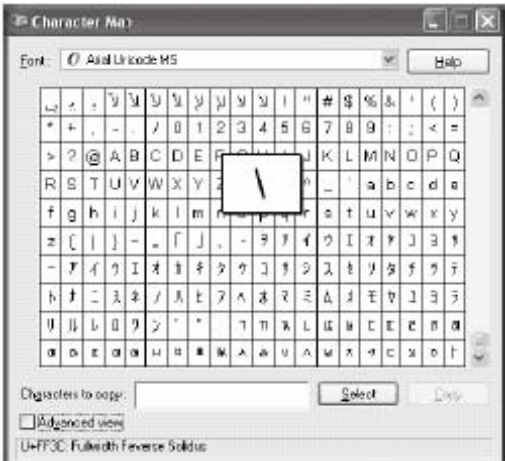
0xFC 0x80 0x80 0x80 0x80 0xBF A bad UTF-8 parser might determine that all of these formats are the same, when, in fact, only 0x3F is valid. Perhaps the most famous UTF-8 attack was against unpatched Microsoft Internet Information Server (IIS) 4 and IIS 5 servers. If an attacker made a request that looked like this— <http://servername/scripts/..%c0%af../winnt/system32/cmd.exe>—the server didn't correctly handle %c0%af in the URL. What do you think %c0%af means? It's 11000000 10101111 in binary; and if it's broken up using the UTF-8 mapping rules in Table 12-2, we get this: **11000000 10101111**. Therefore, the character is

00000101111, or 0x2F, the slash (/) character! The %c0%af is an invalid UTF-8 representation of the / character. Such an invalid UTF-8 escape is often referred to as an *overlong sequence*. So when the attacker requested the tainted URL, he accessed `http://servername/scripts/../../winnt/system32/cmd.exe`. In other words, he walked out of the script's virtual directory, which is marked to allow program execution, up to the root and down into the system32 directory, where he could pass commands to the command shell, Cmd.exe.

**MORE INFO**

You can read more about the "File Permission Canonicalization" vulnerability at [www.microsoft.com/technet/security/bulletin/MS00-057.asp](http://www.microsoft.com/technet/security/bulletin/MS00-057.asp). *UCS-2 Unicode Encoding*

UCS-2 issues are a variation of hex encoding and, to some extent, UTF-8 encoding. Two-byte Universal Character Set, UCS-2, can be hex-encoded in a similar manner as ASCII characters but with the %uNNNN format, where NNNN is the hexadecimal value of the Unicode character. For example, %5C is the ASCII and UTF-8 hex escape for the backslash (\) character, and %u005C is the same character in two-byte Unicode. To really confuse things, %u005C can also be represented by a wide Unicode equivalent called a *fullwidth* version. The fullwidth encodings are provided by Unicode to support conversions between some legacy Asian double-byte encoding systems. The characters in the range %uFF00 to %uFFEF are reserved as the fullwidth equivalents of %20 to %7E. For example, the \ character is %u005C and %uFF3C. You can view these characters by using the Character Map application included with Microsoft Windows. Figure 12-1 shows the backslash character once the Arial Unicode MS font is installed from Microsoft Office XP.



**Figure 12-1** Using the Character Map application to view Unicode characters. *Double Encoding* Just when you thought you understood the various encoding schemes—and we've looked at only the most common—along comes double encoding, which involves reencoding the encoded data. For example, the UTF-8 escape for the backslash character is %5C, which is made up of three characters—%, 5, and C—all of which can be reencoded using their UTF-8 escapes, %25, %35, and %63. Table 12-3 outlines some double-encoding variations of the \ character.

| Escape    | Comments                                                                     |
|-----------|------------------------------------------------------------------------------|
| %5C       | Normal UTF-8 escape of the backslash character                               |
| %255C     | %25, the escape for % followed by 5C                                         |
| %%35%63   | The % character followed by %35, the escape for 5, and %63, the escape for C |
| %25%35%63 | The individual escapes for %, 5, and C                                       |

The vulnerability lies in the mistaken belief that a simple unescape operation will yield clean, raw data. The application then makes a security decision based on the data, but the data might not be fully unescaped. *HTML Escape Codes* HTML pages can also escape characters by using special characters. For example, angle brackets (< and >) can be represented as &lt; and &gt;, and the pounds sterling symbol can be represented as &pound;. But wait, there's more! These escape sequences can also be represented using the

decimal or hexadecimal character values, not just easy-to-remember mnemonics, such as &lt; (less than) and &gt; (greater than). For example, &lt; is the same as &#x3C; (hexadecimal value of the < character) and is also the same as &#60; (decimal value of the < character). A complete list of these entities is available at [www.w3.org/TR/REC-html40/sgml/entities.html](http://www.w3.org/TR/REC-html40/sgml/entities.html). As you can see, many ways exist to encode data on the Web, which makes making decisions based on the name of a resource a dangerous programming practice. Let's now focus on remedies for these issues. *Web-Based Canonicalization Remedies* Like all potential canonicalization vulnerabilities, the first defense is simply not to make decisions based on the name of a resource if it's possible to represent the resource name in more than one way.

### Restrict What Is Valid Input

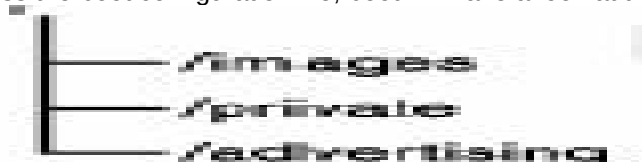
The next best remedy is to restrict what is a valid user request. You created the resources being protected, so you can define the valid ways to access that data and reject all other requests. This is achieved using regular expressions, which are discussed in Chapter 8. Learning to define and use good regular expressions is critical to the security of your application. I'll say it just one more time: always determine what is valid input and reject all other input. It's safer to have a client complain that something doesn't work because of an over-zealous regular expression, than have the service not work because it's been hacked!

### Be Careful When Dealing with UTF-8

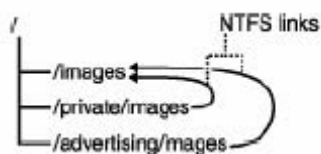
If you must manipulate UTF-8 characters, you need to reduce the data to its canonical form by using the *MultiByteToWideChar* function in Windows. The following sample code shows how you can call this function with various valid and invalid UTF-8 characters. You can find the complete code listing on the companion CD in the folder Secureco\Chapter 12\UTF8. Also note that if you want to create UTF-8 characters, you can use *WideCharToMultiByte* by setting the code page to *CP\_UTF8*. void FromUTF8(LPBYTE pUTF8, DWORD cbUTF8) { WCHAR wszResult[MAX\_CHAR+1]; DWORD dwResult = MAX\_CHAR; int iRes = MultiByteToWideChar(CP\_UTF8, 0, (LPCSTR)pUTF8, cbUTF8, wszResult, dwResult); if (iRes == 0) { DWORD dwErr = GetLastError(); printf("MultiByteToWideChar() failed - > %d\n", dwErr); } else { printf("MultiByteToWideChar() returned " "%s (%d) wide characters\n", wszResult, iRes); } } § void main() { // Get Unicode for 0x5c; should be '\'. BYTE pUTF8\_1[] = {0x5C}; DWORD cbUTF8\_1 = sizeof pUTF8\_1; FromUTF8(pUTF8\_1, cbUTF8\_1); // Get Unicode for 0xC0 0xAF. // Should fail because this is// an overlong '\'. BYTE pUTF8\_2[] = {0xC0, 0xAF}; DWORD cbUTF8\_2 = sizeof pUTF8\_2; FromUTF8(pUTF8\_2, cbUTF8\_2); // Get Unicode for 0xC2 0xA9; should be// a '©' symbol. BYTE pUTF8\_3[] = {0xC2, 0xA9}; DWORD cbUTF8\_3 = sizeof pUTF8\_3; FromUTF8(pUTF8\_3, cbUTF8\_3); }

### Design "Parent Paths" Out of Your Application

Another canonicalization issue relates to the handling of parent paths (..), which can lead to directory traversal issues if not done correctly. You should design your Web-based system in such a way that parent paths are not required when data within the application is being accessed. It's common to see a Web application with a directory structure that requires the use of parent paths, thereby encouraging attackers to attempt to access data outside of the Web root by using URLs like <http://servername/../../boot.ini> to access the boot configuration file, boot.ini. Take a look at the example directory structure in Figure 12-2.



As you can see, a common source of images is used throughout the application. To access an image file from a directory that is below the images directory or that is a peer of the images directory—for example, advertising and private—your application will need to move out of the current directory into the images directory, therefore requiring that your application use parent paths. For example, to load an image, a file named /private/default.aspx would need to use <IMG> tags that look like this: <IMG SRC=../images/Logo.jpg> However, in Windows 2000 and later, the need for parent paths can be reduced. You can create a junction point to the images directory or a hard link to an individual file in the images directory from within the present directory. Figure 12-3 shows what the newer directory structure looks like. It's more secure because there's no need to access any file or directory by using parent paths; your application can remove multiple dots as a requirement in a valid file request.



**Figure 12-3** A common Web application directory structure using links to a parent or peer directory. With this directory format in place, the application can access the image without using parent paths, like so:<IMG SRC=images/Logo.jpg> You can create

junction points by using the Linkd.exe tool included in the Windows 2000 Resource Kit, and you can link to an individual file by using the *CreateHardLink* function. The following is a simple example of using the *CreateHardLink* function to create hard links to files. You can also find this example code on the companion CD in the folder Secureco\Chapter12\HardLink./\* HardLink.cpp\*/ #include "stdafx.h" DWORD DoHardLink(LPCSTR szName, LPCSTR szTarget) { DWORD dwErr = 0; if (!CreateHardLink(szName, szTarget, NULL)) dwErr = GetLastError(); return dwErr;} void main(int argc, char\* argv[]) { if (argc != 3) { printf("Usage: HardLink <linkname> <target>\n") ; } DWORD dwErr = DoHardLink(argv[1], argv[2]); if (dwErr) printf("Error calling CreateHardLink() - > %d\n", dwErr); else printf("Hard link created to %s\n", argv[2]); }

## NOTE

Just say no to parent paths. If you remove the requirement for parent paths in your application, anyone attempting to access a resource by using parent paths is, by definition, an attacker!

## Question: 60

You are an application developer for Company.com You are modifying an application that was developed by another developer. The application was developed by using the Microsoft .NET Framework. The application accepts user input into a variable named userInput and saves information in a Microsoft SQL Server database.

The application uses the following code to construct a SQL query string sqlquery sqlquery = "SELECT FROM Table1 WHERE ColumnA = " + userInput = "";

You need to improve the security of the application to reduce the likelihood of SQL injection attacks from user input.

Which three actions should you perform for all user input? (Each correct answer presents part of the solution. Choose three)

- A. Remove double hyphens (--) from the input.
- B. Encode the input by using the HttpUtility.HtmlEncode method.
- C. Decode the input by using the HttpUtility.HtmlDecode method.
- D. Limit the length to the exact size of the SQL Server column where the input will be stored.
- E. Replace single quotation marks (') with two single quotation marks (").
- F. Convert all characters to Unicode.

**Answer: A, D, E**

## Explanation

**Preventing SQL Injection Attacks** If you design your scripts and applications with care, SQL injection attacks can be avoided most of the time. There are a number of things that we as developers can do to reduce our site's susceptibility to attack. Here's a list (in no particular order) of our options:

### Limit User Access

The default system account (sa) for SQL server 2000 should never be used because of its unrestricted nature. You should always setup specific accounts for specific purposes. For example, if you run a database that lets users of your site view and order products, then you should set up a user called webUser\_public that has SELECT rights on the products table, and INSERT rights only on the orders table. If you don't make use of extended stored procedures, or have unused triggers, stored procedures, user-defined functions, etc, then remove them, or move them to an isolated server. Most extremely damaging SQL injection attacks attempt to make use of several extended stored procedures such as xp\_cmdshell and xp\_grantlogin, so by removing them, you're theoretically blocking the attack before it can occur.

### Escape Quotes

As we've seen from the examples discussed above, the majority of injection attacks require the user of single quotes to terminate an expression. By using a simple replace function and converting all single quotes to two single quotes, you're greatly reducing the chance of an injection attack succeeding. Using ASP, it's a simple matter of creating a generic replace function that will handle the single quotes automatically, like this:

```
<%function stripQuotes(strWords) stripQuotes = replace(strWords, "'", "''") end function %>
```

Now if we use the stripQuotes function in conjunction with our first query for example, then it would go from this:

```
select count(*) from users where userName='john' and userPass=" or 1=1 --'
```

...to this:

select count(\*) from users where userName='john' and userPass='' or 1=1 --' This, in effect, stops the injection attack from taking place, because the clause for the WHERE query now requires both the userName and userPass fields to be valid.

### ***Remove Culprit Characters/Character Sequences***

As we've seen in this article, certain characters and character sequences such as ;, --, select, insert and xp\_ can be used to perform an SQL injection attack. By removing these characters and character sequences from user input before we build a query, we can help reduce the chance of an injection attack even further. As with the single quote solution, we just need a basic function to handle all of this for us:

```
<%function killChars(strWords) dim badChars dim newChars badChars = array("select", "drop", ";", "--", "insert", "delete", "xp_")
newChars = strWords for i = 0 to uBound(badChars) newChars = replace(newChars, badChars(i), "") next killChars = newChars end
function %>
```

Using stripQuotes in combination with killChars greatly removes the chance of any SQL injection attack from succeeding. So if we had the query:

```
select prodName from products where id=1; xp_cmdshell 'format
```

```
c: /q /yes '; drop database myDB; -- and ran it through stripQuotes and then killChars, it would end up looking like this: prodName
from products where id=1 cmdshell "format c: /q /yes " database myDB
...which is basically useless, and will return no records from the query.
```

### ***Limit the Length of User Input***

It's no good having a text box on a form that can accept 50 characters if the field you'll compare it against can only accept 10. By keeping all text boxes and form fields as short as possible, you're taking away the number of characters that can be used to formulate an SQL injection attack. If you're accepting a querystring value for a product ID or the like, always use a function to check if the value is actually numeric, such as the IsNumeric() function for ASP. If the value isn't numeric, then either raise an error or redirect the user to another page where they can choose a product. Also, always try to post your forms with the method attribute set to POST, so clued-up users don't get any ideas --- they might if they saw your form variables tacked onto the end of the URL.

## **SQL Server 2000 SP3 Security Features and Best Practices: Security Best Practices Checklist**

Administrator Checklist Developer Checklist In addition to all of the items above, the following should be considered best practices for developers.

| Setting Up the Environment Prior to Installation |                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Physical security                                | <ul style="list-style-type: none"> <li>• Ensure the physical security of your server.</li> </ul>                                                                                                                                                                                                                                                                                                |
| Firewalls                                        | <ul style="list-style-type: none"> <li>• Put a firewall between your server and the Internet.</li> <li>• Always block TCP port 1433 and UDP port 1434 on your perimeter firewall. If named instances are listening on additional ports, block those too.</li> <li>• In a multi-tier environment, use multiple firewalls to create screened subnets.</li> </ul>                                  |
| Isolation of services                            | <ul style="list-style-type: none"> <li>• Isolate services to reduce the risk that a compromised service could be used to compromise others.</li> <li>• Never install SQL Server on a domain controller.</li> <li>• Run separate SQL Server services under separate Windows accounts.</li> <li>• In a multi-tier environment, run Web logic and business logic on separate computers.</li> </ul> |
| Service accounts                                 | <ul style="list-style-type: none"> <li>• Create Windows accounts with the lowest possible privileges for running SQL Server services.</li> </ul>                                                                                                                                                                                                                                                |
| File System                                      | <ul style="list-style-type: none"> <li>• Use NTFS.</li> <li>• Use RAID for critical data files.</li> </ul>                                                                                                                                                                                                                                                                                      |

| Installation                    |                                                                                                                                                                                                                           |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Latest version and service pack | <ul style="list-style-type: none"> <li>• Always install the latest service packs and security patches.</li> </ul>                                                                                                         |
| Service accounts                | <ul style="list-style-type: none"> <li>• Run SQL Server services with the lowest possible privileges.</li> <li>• Use Enterprise Manager to associate services with Windows accounts.</li> </ul>                           |
| Authentication mode             | <ul style="list-style-type: none"> <li>• Require Windows Authentication for connections to SQL Server.</li> </ul>                                                                                                         |
| Strong passwords                | <ul style="list-style-type: none"> <li>• Always assign a strong password to the <b>sa</b> account, even when using Windows Authentication.</li> <li>• Always use strong passwords for all SQL Server accounts.</li> </ul> |

| Configuration Options and Settings After Installation |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Delete or secure old setup files                      | <ul style="list-style-type: none"> <li>Delete or archive the following files after installation: sqlstp.log, sqlsp.log, and setup.iss in the &lt;systemdrive&gt;\Program Files\Microsoft SQL Server\MSSQL\Install folder for a default installation, and the &lt;systemdrive&gt;\Program Files\Microsoft SQL Server\MSSQL\$&lt;Instance Name&gt;\Uninstall folder for named instances.</li> <li>If the current system is an upgrade from SQL Server 7.0, delete the following files: setup.iss in the %Windir% folder, and sqlsp.log in the Windows Temp folder.</li> </ul> |
| Choose static ports for named instances               | <ul style="list-style-type: none"> <li>Assign static ports to named instances of SQL Server.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Set login auditing level                              | <ul style="list-style-type: none"> <li>Set login auditing level to <b>failure</b> or <b>all</b>.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Enable security auditing                              | <ul style="list-style-type: none"> <li>Enable security auditing of <b>Sysadmin</b> actions, fixed role membership changes, all login related activity, and password changes.</li> <li>After selecting appropriate auditing options, you should script the audit, wrap it in a stored procedure, and mark that stored procedure for AutoStart.</li> </ul>                                                                                                                                                                                                                    |
| Secure sa even in Windows Authentication Mode         | <ul style="list-style-type: none"> <li>Assign a strong password to the <b>sa</b> account, even on servers that are configured to require Windows Authentication.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                 |
| Remove sample databases                               | <ul style="list-style-type: none"> <li>Remove sample databases from production servers.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

| Secure Operation                  |                                                                                                                                                                                                                                                                             |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Security model                    | <ul style="list-style-type: none"> <li>• Learn to work with the SQL Server security model.</li> </ul>                                                                                                                                                                       |
| Backup policy                     | <ul style="list-style-type: none"> <li>• Back up all data regularly and store copies in a secure off-site location.</li> <li>• Test your disaster recovery system.</li> </ul>                                                                                               |
| Surface and feature reduction     | <ul style="list-style-type: none"> <li>• Reduce the surface area of your system that is exposed to attack by running only those services and features needed in your environment.</li> </ul>                                                                                |
| Administrator reduction           | <ul style="list-style-type: none"> <li>• Restrict membership of the <b>sysadmin</b> fixed server role to a few trusted individuals.</li> </ul>                                                                                                                              |
| Strong passwords                  | <ul style="list-style-type: none"> <li>• Ensure that you use complex passwords for all SQL Server accounts.</li> </ul>                                                                                                                                                      |
| Cross database ownership chaining | <ul style="list-style-type: none"> <li>• Disable cross database ownership chaining if your system does not use it.</li> </ul>                                                                                                                                               |
| Xp_cmdshell                       | <ul style="list-style-type: none"> <li>• By default, only members of the sysadmin role can execute xp_cmdshell. You should not change this default.</li> <li>• Do not grant execute permission on xp_cmdshell to users who are not members of the sysadmin role.</li> </ul> |

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Encryption          | <ul style="list-style-type: none"> <li>• Install a certificate to enable SSL connections.</li> <li>• Certificates should use the fully-qualified DNS name of the server.</li> <li>• Use the SQL Server service account to encrypt database files with EFS.</li> <li>• If your application requires data encryption, consider using the products of such vendors as Protegrity and Application Security Inc.</li> </ul>                                                      |
| Roles and groups    | <ul style="list-style-type: none"> <li>• Collect users into SQL Server roles or Windows groups to simplify permissions administration.</li> </ul>                                                                                                                                                                                                                                                                                                                           |
| Permissions         | <ul style="list-style-type: none"> <li>• Never grant permissions to the <b>public</b> database role.</li> </ul>                                                                                                                                                                                                                                                                                                                                                             |
| Distributed queries | <ul style="list-style-type: none"> <li>• When setting up SQL Server in an environment that supports distributed queries, use linked servers rather than remote servers.</li> <li>• Allow linked server access only to those logins that need it.</li> <li>• Disable ad hoc data access on all providers except SQL OLE DB, for all users except members of the <b>sysadmin</b> fixed server role.</li> <li>• Allow ad hoc data access only on trusted providers.</li> </ul> |
| Guest accounts      | <ul style="list-style-type: none"> <li>• Do not enable the guest account.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                        |
| Service accounts    | <ul style="list-style-type: none"> <li>• If you need to change the account associated with a SQL Server service, use SQL Server Enterprise Manager.</li> <li>• If you change multiple services, you must apply the changes to each service separately using Enterprise Manager.</li> </ul>                                                                                                                                                                                  |

| Best Practices for Patching Instances |                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Instance detection and enumeration    | <ul style="list-style-type: none"> <li>• Keep an inventory of all versions, editions, and languages of SQL Server for which you are responsible.</li> <li>• Include instances of MSDE in your inventory.</li> <li>• Use SQL Scan and SQL Check, available from the Microsoft Web site, to scan for instances of SQL Server within your domain.</li> </ul> |
| Bulletins                             | <ul style="list-style-type: none"> <li>• Subscribe to Microsoft security bulletins.</li> </ul>                                                                                                                                                                                                                                                            |
| Patch application                     | <ul style="list-style-type: none"> <li>• Maintain test systems that match the configuration of your production systems, and are readily available for testing new patches.</li> <li>• Test patches carefully before applying them to production systems.</li> <li>• Consider patching development systems with relatively little testing.</li> </ul>      |

| Recommended Periodic Administrative Procedures |                                                                                                                                                                                                                                                                              |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Microsoft Baseline Security Analyzer           | <ul style="list-style-type: none"> <li>• Add MBSA to your weekly maintenance schedule, and follow up on any security recommendations that it makes.</li> </ul>                                                                                                               |
| Scanning logins                                | <ul style="list-style-type: none"> <li>• Periodically scan for accounts with NULL passwords and remove them or assign them strong passwords.</li> <li>• Delete unused accounts.</li> </ul>                                                                                   |
| Enumerate fixed role membership                | <ul style="list-style-type: none"> <li>• Periodically scan fixed server and database roles to ensure that membership is only granted to trusted individuals.</li> </ul>                                                                                                      |
| Start-up procedures                            | <ul style="list-style-type: none"> <li>• Verify the safety of stored procedures that have been marked for AutoStart.</li> </ul>                                                                                                                                              |
| Login-to-user mapping                          | <ul style="list-style-type: none"> <li>• Ensure that the mapping between database users and logins at the server level is correct.</li> <li>• Run <b>sp_change_users_login</b> with the <b>report</b> option regularly to ensure that the mapping is as expected.</li> </ul> |
| Direct catalog updates                         | <ul style="list-style-type: none"> <li>• Do not allow direct catalog updates.</li> </ul>                                                                                                                                                                                     |
| Cross database ownership chaining              | <ul style="list-style-type: none"> <li>• Use <b>sp_dboption</b> to enumerate and validate databases for which cross database ownership chaining has been enabled.</li> </ul>                                                                                                 |

| General                                                   |                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Use ownership chaining effectively                        | <ul style="list-style-type: none"> <li>• Use ownership chaining within a single database to simplify permissions management.</li> <li>• Avoid using cross database ownership chaining when possible.</li> <li>• If you must use cross database ownership chaining, ensure that the two databases are always deployed as a single administrative unit.</li> </ul> |
| Use roles to simplify permission management and ownership | <ul style="list-style-type: none"> <li>• Assign permissions to roles rather than directly to users.</li> <li>• Objects may be owned by roles, rather than directly by users, if you want to avoid application changes when the owning user is dropped.</li> </ul>                                                                                                |
| Turn on encryption (SSL or IPSEC)                         | <ul style="list-style-type: none"> <li>• Enable encrypted connections to your server, and consider allowing only encrypted connections.</li> <li>• When allowing SQL Server Authentication, you are strongly urged to encrypt either the network layer with IPsec or the session with SSL.</li> </ul>                                                            |
| Do not propagate SQL Server errors back to user           | <ul style="list-style-type: none"> <li>• Your application should not return SQL Server errors to the end user. Log them instead, or transmit them to the system administrator.</li> </ul>                                                                                                                                                                        |
| Prevent SQL injection                                     | <ul style="list-style-type: none"> <li>• Defend against SQL injection by validating all user input before transmitting it to the server.</li> <li>• Limit the scope of possible damage by permitting only minimally privileged accounts to send user input to the server.</li> <li>• Run SQL Server itself with the least necessary privileges.</li> </ul>       |

| Multi-tier Options                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Same/trusted domain (complete Windows Authentication) | <p>If the application server and the database server are within the same domain, or within trusted domains, you should use Windows Authentication and configure for 'full provisioning' in which all client contexts are tunneled to SQL Server. This makes it possible to audit all users who access SQL Server, enables Windows security policy enforcement, and makes it unnecessary to store credentials in the middle tier. In this scenario, the client connects to the application server, which in turn impersonates the client and connects to SQL Server.</p> <ul style="list-style-type: none"> <li>• Every user on the application server must have a valid Windows login on the database server and delegation must be enabled.</li> <li>• All systems interacting in this scenario, including the Domain Controller, must run Windows 2000 or higher.</li> <li>• The account the application is running under must be trusted for delegation (that is, the Active Directory option <b>Account is trusted for delegation</b> must be turned on for this account).</li> <li>• The client account must be able to be delegated (ensure that the Active Directory user account option <b>Account is trusted and cannot be delegated</b> is unchecked).</li> <li>• The application service must have a valid Service Principal Name (SPN).</li> </ul> <p><b>Note:</b> Full provisioning is not recommended in cross-enterprise or Internet-scale installations, when your security plan calls for minimizing user access to the database server, or in enterprises with policies prohibiting delegation.</p> |

|                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mixed scenario (partial Windows Authentication)                         | <p>When the Internet-facing tier does not have an individual Windows domain account for every possible user, the recommended scenario is to divide authentication into stages. The outer tier (which authenticates users) should use SSL to encrypt at least credentials, if not the entire session. It should connect to the database server using Windows Authentication, forwarding transaction information under a separate security context that is low privileged, with only the permissions necessary to perform its function. This effectively uses the middle tier as an additional layer of defense between your server and the Internet.</p> <p><b>Note:</b> Using SQL Server Authentication between the middle tier and SQL Server is not recommended, because of the need to store credentials. If you must use SQL Server Authentication between the middle tier and SQL Server, you should create several accounts, with different levels of privileges corresponding to different classes of users. This requires that you add logic to the middle tier to allocate connections according to the desired privilege level.</p> |
| Different non-trusted domains or no domains (no Windows Authentication) | <p>When Windows Authentication between tiers is not possible, you should require SSL encryption of the login sequence. Encrypting the entire session is preferable.</p> <ul style="list-style-type: none"> <li>• You should also use DPAPI to encrypt credentials that must be stored.</li> <li>• You should store encrypted credentials in a registry key protected with an ACL.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

Software Vendor Checklist In addition to all of the items above, the following security development practices have proven useful in increasing the quality and security of code in various development environments.

## Security Processes

Understanding various security issues

- Ensure that members of your development team understand major security issues: current threats, security trends, changing security environments, and attack scenarios.
- Require relevant security training for all developers and testers.
- Increase the awareness of issues like cross-site scripting, buffer overflows, SQL injection, and dangerous APIs.
- Identify specific categories of threats that apply to your product — for example, denial of service, escalation of privileges, spoofing, data tampering, information disclosure and repudiation.
- Analyze security threats to your product, component-by-component.
- Create a security threat checklist based on your product.
- Add security reviews to all stages (from design to testing) of your product development cycle.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MSDE installations | <p>If you distribute MSDE with your application, the following additional guidance applies:</p> <ul style="list-style-type: none"> <li>• Install MSDE using "Windows security mode" as the default.</li> <li>• Never install a blank sa password.</li> <li>• When distributing MSDE to your customers, you should use the Microsoft-supplied installer rather than merge modules.</li> <li>• When installing an instance of MSDE that will operate only as a local data store, you should disable the Server Net-Libraries.</li> <li>• If your product includes MSDE, you should make this known to your customers. In the future, they may need to install or accept MSDE-specific software updates.</li> <li>• MSDE installs SQL Server Agent by default, but leaves the Service startup type to "Manual." If your application does not use SQL Server Agent, you should change this to "Disabled." Include security best practice information in your product documentation.</li> </ul> |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Question: 61

You are an application developer for Company.com A developer in Company develops a component named CompanyComponent. You deploy CompanyComponent to a server. After you execute

CompanyComponent, you receive an error message that reads in part:  
"System.Security.Policy.PolicyException."

You need to find out the cause of the error.

What should you do?

- A. View the NTFS permissions of MyComponent.dll to find out the required permissions.
- B. View the permissions requested by the component at run time by using the Microsoft CLR Debugger.
- C. View the permissions requested by the component at run time by using the Runtime Debugger.
- D. View the required permissions for the component by using the Permissions View tool.

**Answer: D**

### Explanation PolicyException Class

The exception that is thrown when policy forbids code to run. For a list of all members of this type, see PolicyException Members.  
System.Object System.Exception System.SystemException

## System.Security.Policy.PolicyException

<Serializable>

### Public Class PolicyException Inherits SystemException

#### Remarks

This exception is typically thrown when the code requests more permissions than the policy will grant or the policy is configured to prohibit running the code

#### Permissions View Tool (Permview.exe)

The Permissions View tool is used to view the minimal, optional, and refused permission sets requested by an assembly. Optionally, you can use Permview.exe to view all declarative security used by an assembly. **permview** [/output *filename*] [/decl] *manifestfile*

| Argument                | Description                                                                                                                                                                                                                                                                                                                             |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>manifestfile</i>     | The file that contains the assembly's manifest. The manifest can be either a standalone file or it can be incorporated in a portable executable (PE) file. The extension for this file will usually be .exe or .dll, but it could also be .scr, or .ocx.                                                                                |
| Option                  | Description                                                                                                                                                                                                                                                                                                                             |
| /decl                   | Displays all declarative security at the assembly, class, and method level for the assembly specified by <i>manifestfile</i> . This includes permission requests as well as demands, asserts, and all other security actions that can be applied declaratively. It does not refer to other assemblies linked to the specified assembly. |
| /h[elp]                 | Displays command syntax and options for the tool.                                                                                                                                                                                                                                                                                       |
| /output <i>filename</i> | Writes the output to the specified file. The default is to display the output to the console.                                                                                                                                                                                                                                           |
| /?                      | Displays command syntax and options for the tool.                                                                                                                                                                                                                                                                                       |

#### Remarks

Developers can use Permview.exe to verify that they have applied permission requests correctly to their code. Additionally, users can run Permview.exe to determine the permissions an assembly requires to execute. For example, if you run a managed executable and get the error, "System.Security.Policy.PolicyException: Failed to acquire required permissions," you can use Permview.exe to determine the permissions the code in your executable must receive before it will execute.

#### Examples

The following command displays the permissions requested by the assembly myAssembly.exe to the console. permview myAssembly.exe If myAssembly.exe contains a minimum request for **FullTrust**, the following output appears. Microsoft (R) .NET Framework Permission Request Viewer. Version 1.0.2204.18 Copyright (C) Microsoft Corp. 1998-2000 minimal permission set: <PermissionSet class="System.Security.PermissionSet" version="1"> <Unrestricted/></PermissionSet> optional permission set: Not specified refused permission set: Not specified

The following command displays all declarative security on the assembly myAssembly.exe to the console. This command displays the method level security demand. permview /decl myAssembly.exe The following output appears. Microsoft (R) .NET Framework Permission Request Viewer. Version 1.0.2204.18 Copyright (C) Microsoft Corp. 1998-2000 Assembly Request Minimum permission set: <PermissionSet class="System.Security.PermissionSet" version="1"> <Unrestricted/></PermissionSet> Method A::myMethod() LinktimeCheck permission set: <PermissionSet class="System.Security.PermissionSet" version="1"><Permission class="System.Security.Permissions.ReflectionPermission, mscorlib, Ver=1.0.2204.2, Loc=", SN=03689116d3a4ae33" version="1"> <MemberAccess/> </Permission> </PermissionSet> The following command writes the permissions requested by the assembly myAssembly.exe to the file myOutputFile. permview /output myOutputFile myAssembly.exe

#### Question: 62

You are an application developer for Company.com. You develop an assembly for a Windows-based application that is installed on all computers in Company. A written company policy states that access to the assembly is not allowed from computers at one office named Location1. Access to the assembly is allowed from all other locations in the company. One employee at Location1 is granted an exception to this policy and requires access to the assembly.

You need to enable the one employee to access the assembly. You plan to achieve this goal by using the Microsoft .NET Framework

Configuration tool.

What should you do?

- A. Grant the FullTrust permission for the assembly at the enterprise level. Deny all permissions for the assembly at the user level.
- B. Grant the FullTrust permission for the assembly at the enterprise level and the user level. Deny all permissions for the assembly at the machine level.
- C. Grant the FullTrust permission for the assembly at the machine level. Deny all permissions for the assembly at the user level and the enterprise level.
- D. Grant the FullTrust permission for the assembly at the machine level and the user level. Deny all permissions for the assembly at the enterprise level.

**Answer: B**

### **Explanation User Policy Administration**

User policy is the lowest administrable policy level. Every user has an individual user policy configuration file. Any changes made to this policy level are applicable only to the current logged-on user. The user policy level is restricted in what it can specify. Because this level is configurable by the current logged-on user, enterprise level policy administrators should be aware that the user might potentially alter any policy changes made on the user policy level. The user policy level is not able to give more permissions to an assembly than is specified in the higher policy levels. However, the user policy level is allowed to decrease permissions, which might potentially cause applications to stop functioning properly. If the **LevelFinal** attribute is applied to a code group on the machine or enterprise level, the user level is not allowed to tighten policy decisions that have been made on those levels. User level administration is appropriate in some situations to tightening security. For example, a user might decide to tighten security policy for assemblies that originate from the local intranet zone if untrusted code is found there. You might consider administering policy on this level when you are a user on a corporate network and believe that the security settings are not tight enough.

### **Machine Policy Administration**

The machine policy level holds most of the default security policy. All machine and domain administrators have access to the machine configuration files. Machine administrators can set policy that excludes modification from the user level but not from the enterprise level. You might consider administering security policy on this level in the following situations:

You are not on a network or are on a network without a domain controller.

The computer you are administering serves a unique function. For example, if you are administering a public computer that is used for general Internet access by several people in a semi-public setting, you might want to have a unique machine policy, because the computer serves a unique function. Additionally, you might want to produce a specific machine policy that considers the security needs of specialized computers, like the servers in your enterprise.

### **Enterprise Policy Administration**

The enterprise policy level affects every computer and user on the network and can only be administered by enterprise or domain administrators. See the section on Deploying Security Policy for information on deployment strategies. Because the runtime evaluates enterprise policy first, you can apply the LevelFinal attribute to a code group on this level to exclude the lower levels from making policy changes. If you do not apply the LevelFinal attribute to code groups on this level, administrators of lower security levels will be able to assign more permissions to applications without your knowledge and potentially create security vulnerabilities. You might consider administering policy on this level when every person in your enterprise uses an application and you want to make sure that it always receives sufficient permission to run.

### **Question: 63**

You are an application developer for Company.com You are developing a library assembly that communicates with a hardware device. Users will use the library assembly and the hardware device to develop custom applications. Users want their applications to be able to control access to the hardware device by using code access security policies.

You need to ensure that users can code access security to restrict access to the hardware device.

What should you do?

- A. Create a custom permissions class. Modify the library installer to install this permission and a new permission set.
- B. Create a custom evidence class.
- C. Create a custom security policy and a security policy development package. Install this package at the same time as the library assembly.
- D. Create a publisher policy assembly for the library assembly. Install this assembly with the library assembly.

**Answer: A**

### **Explanation Implementing a Custom Permission**

All permission objects must implement the `IPermission` interface. Inheriting from the `CodeAccessPermission` class is the easiest way to create a custom permission because `CodeAccessPermission` implements `IPermission` and provides most of the methods required for a permission. Additionally, you must implement the `IUnrestrictedPermission` interface for all custom code access permissions. The custom permission class is required for both imperative and declarative security support, so you should create it even if you plan to use only declarative security. Defining the Permission Class To derive from the `CodeAccessPermission` class, you must override the following five key methods and provide your own implementation:

**Copy** creates a duplicate of the current permission object.

**Intersect** returns the intersection of allowed permissions of the current class and a passed class.

**IsSubsetOf** returns **true** if a passed permission includes everything allowed by the current permission.

**FromXml** decodes an XML representation of your custom permission.

**ToXml** encodes an XML representation of your custom permission. The **IUnrestrictedPermission** interface requires you to override and implement a single method called **IsUnrestrictedPermission**. In order to support the **IUnrestrictedPermission** interface, you must implement some system, such as a Boolean value that represents the state of restriction in the current object, to define whether the current instance of the permission is unrestricted.

### **Creating Your Own Code Access Permissions**

The .NET Framework supplies a set of code access permission classes designed to help protect a specific set of resources and operations, focusing on those resources exposed by the .NET Framework. These permission classes are described briefly in the Permissions topic and in detail in the reference documentation for each permission class. For most environments, the built-in code access permissions are adequate. However, in some situations, it might make sense to define your own code access permission class. This topic discusses when, why, and how to define custom code access permission classes. If you are defining a component or class library that accesses a resource that is not covered by the built-in permission classes but needs to be protected from unauthorized code, you should consider creating a custom code access permission class. If you want to be able to make declarative demands for your custom permission, you must also define an attribute class for the permission. Providing these classes and making demands for the permission from within your class library enables the runtime to prevent unauthorized code from accessing that resource and enables an administrator to configure access rights. There are other situations in which a custom permission might be appropriate. When a built-in code access permission class protects a resource but does not sufficiently control access to that resource, you might need a custom code access permission. For example, an application might use personnel records for which each employee record is stored in a separate file; in such a case, read and write access could be controlled independently for different types of employee data. An internal management tool could be authorized to read certain sections of an employee's personnel file but not to modify those sections. In fact, it might not even be allowed to read some sections. Custom code access permissions are also appropriate in cases where a built-in permission exists but is not defined in a way that enables it to protect the resource appropriately. For example, there might be a case in which there is UI functionality, such as the ability to create menus, that must be protected but is not protected by the built-in `UIPermission` class. In that case, you could create a custom permission to protect the ability to create menus. Wherever possible, permissions should not overlap. Having more than one permission protecting a resource presents a significant problem for administrators, who must then be sure to deal appropriately with all the overlapping permissions every time they configure the rights to access that resource. Implementing a custom code access permission involves the following steps, some of which are optional. Each step is described in a separate topic.

- 1 Design the Permission class.
- 2 Implement the `IPermission` and `IUnrestrictedPermission` interfaces.
- 3 Implement the `ISerializable` interface, if necessary for performance or to support special data types.
- 4 Handle XML encoding and decoding.
- 5 Add support for declarative security, by implementing an **Attribute** class.

- 6 Demand custom permission for your permission, where appropriate.
- 7 Update security policy to be aware of the custom permission.

**Question: 64**

You are an application developer for Company.com. You are developing a Web service that will allow customers to access confidential information about their accounts. Client applications that use the Web service will provide an identifier and a password. The Web service will verify credentials by using a Microsoft SQL Server database.

You must minimize the risk that account information can be intercepted during transmission.

You need to choose an authentication method.

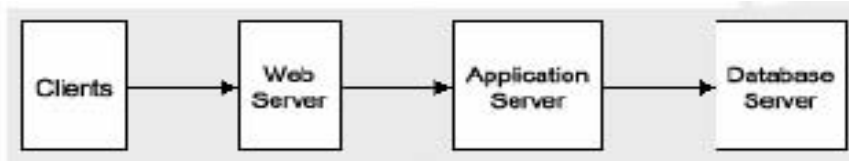
What should you do?

- A. Use Web Service Enhancements for Microsoft .NET (WSE) to implement the WS-Security specification.
- B. Require client applications to encrypt an identifier and password in the SOAP header of all requests.
- C. Use basic authentication over SSL/TLS.
- D. Use Forms authentication over SSL/TLS.

**Answer: D**

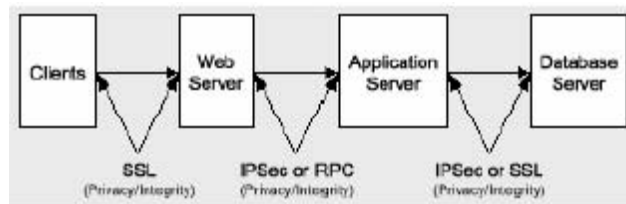
**Explanation**

Secure Sockets Layer / Transport Layer Security (SSL/TLS). This is most commonly used to secure the channel between a browser and Web server. However, it can also be used to secure Web service messages and communications to and from a database server running Microsoft SQL Server 2000. Know What to Secure When a Web request flows across the physical deployment tiers of your application, it crosses a number of communication channels. A



**Figure 1. A typical Web deployment model**

In this typical deployment model, a request passes through three distinct channels. The client-to-Web server link may be over the Internet or corporate intranet and typically uses HTTP. The remaining two links are between internal servers within your corporate domain. Nonetheless, all three links represent potential security concerns. Many purely intranet-based applications convey security sensitive data between tiers; for example, HR and payroll applications that deal with sensitive employee data. Figure 2 shows how each channel can be secured by using a combination of SSL, IPsec and RPC encryption.



**Figure 2. A typical Web deployment model, with secure communications**

The choice of technology depends on a number of factors including the transport protocol, end point technologies, and environmental considerations (such as hardware, operating system versions, firewalls, and so on). SSL/TLS is used to establish an encrypted communication channel between client and server. The handshake mechanism used to establish the secure channel is well documented and details can be found in the following articles in the Microsoft Knowledge Base:

Q257591, "Description of the Secure Sockets Layer (SSL) Handshake"

Q257587, "Description of the Server Authentication Process During the SSL Handshake" Q257586, "Description of the

## Client Authentication Process During the SSL Handshake

### Question: 65

You are an application developer for your company. You are developing an application that needs to exchange a shared key at the start of each communication with remote components. The exchange occurs over the Internet. You need to ensure that only the intended recipient can read the shared key. What should you do?

- A. Sign the shared key by using the application's private key.
- B. Encrypt the shared key by using the application's private key.
- C. Encrypt the shared key by using the remote component's public key.
- D. Encode the shared key by using the `System.Text.Encoding.UTF8` object.
- E. Encode the shared key by using the `System.Text.Encoding.BigEndianUnicode` object.

**Answer: E**

### Question: 66

You are an application developer for Company.com. You develop a library assembly that contains diagnostic utility classes. This library assembly is installed in the global assembly cache on all client computers on Company's network. You develop a Windows Forms application that calls the library assembly. You successfully exam the application on your computer, and then you deploy the application to a Web folder on the intranet. Further examsing reveals that when you run this application from the intranet, a `SecurityException` exception is thrown when the application is loading.

You need to correct the problem that is causing the `SecurityException` exception.

What should you do?

- A. Add the following code segment to the library assembly. [assembly: AllowPartialTrustedCallers]
- B. Add the following code segment to the Windows Forms application assembly. [assembly: AllowPartiallyTrustedCallers]
- C. Add the following code segment to the library assembly. [assembly: PermissionSet(SecurityAction.RequestOptional, Name = "Loca
- D. Add the following code segment to the Windows Forms application assembly. [assembly: PermissionSet(SecurityAction.RequestMinimum, Name = "Local

**Answer: D**

### Explanation

.NET permissions are grouped into `NamedPermissionSets`. The platform includes the following non-modifiable built-in sets: `Nothing`, `Execution`, `FullTrust`, `Internet`, `LocalIntranet`, `SkipVerification`. The `FullTrust` set is a special case, as it declares that this code does not have

any restrictions and passes any permission check, even for custom permissions. By default, all local code (found in the local computer directories) is granted this privilege. The above fixed permission sets can be demanded instead of regular permissions: [assembly:PermissionSetAttribute( SecurityAction.RequestMinimum, Name="LocalIntranet")] Here is a summary of some facts or rules:

1 If you want to restrict the permissions given to an assembly to only those contained in the associated permission set, you must tick the code group option "The policy level will only have the permissions from the permission set associated with this code group". Otherwise what is granted to the assembly is the permissions of the particular associated permission set plus permissions of the associated permission set of the inherited code group ("All\_Code" group).

2 All assemblies must be given "Enable assembly execution" security permission so that it can be run or launched.

3 Permissions included in an assembly's associated permission set that are above the logged-in user's privilege will not be granted.

4 A strongly named assembly can only be called by a fully-trusted caller, unless this assembly states `AllowPartiallyTrustedCallers`. When you use this attribute, it means that you have fully reviewed your code and there is no security flaw that may be used by luring attackers – such as a improperly used `Assert`. Not all system assemblies are marked with this attribute. You can look at the assembly's manifest to see whether it has that attribute.

5 However, an assembly belonging to the root “All\_Code” code group can be called by partially-trusted callers, even if they are strongly named. This is probably because, if you don't impose a particular security control on an assembly, the runtime security thinks that this assembly is not extremely critical.

6 When you states `AllowPartiallyTrustedCallers` in an assembly, or let it stay in the “All\_Code” code group, a permission-checking stack walk is still going to be triggered for every attempt to access any controlled resource. The only difference is if you improperly make a `Assert` you will make luring attacks possible.

### Question: 67

You are an application developer for Company.com, which is a financial services company. You are developing an ASP.NET Web application that will be used by Company's customers. Customers will use the application to access their portfolios and to view business and financial reports. The customers are divided into two categories named Standard and Premier. The Premier customers will have access to an additional set of reports and analysis. You plan to use roles named Standard and Premier to differentiate the two customer categories.

The application will use Forms authentication to authenticate all users and assign each authenticated user to either the Standard role or the Premier role. Web pages that are accessible only be Premier customers are in a subfolder named Premier. Web pages that are accessible by both categories of customers are in the application root.

You need to configure URL authorization for the application. You plan to achieve this goal by adding configuration elements to the `Web.config` file in the application root.

Which elements should you use?

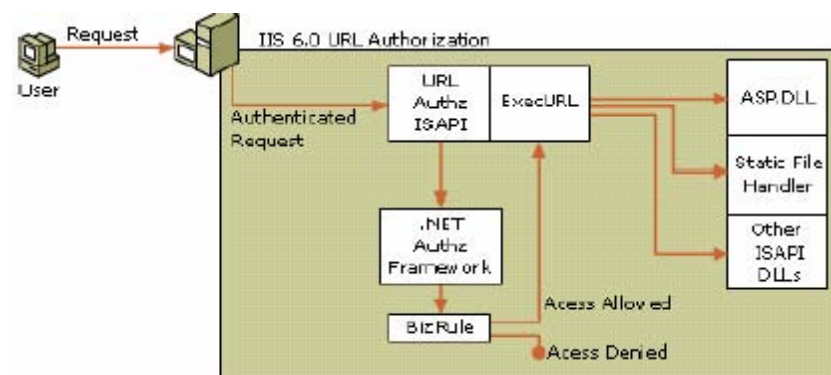
- A. `<authorization> <deny users=?"/> </authorization> <location path="Premier"> <system.web> <authorization> <allow roles="Premier"/> <deny users="*/> </authorization> </system.web> </location>`
- B. `<authorization> <deny users=?"/> </authorization> <location path="Premier"> <system.web> <authorization> <deny users="*/> <allow roles="Premier"/> </authorization> </system.web> </location>`
- C. `<authorization> <deny users=?"/> <deny roles="Premier"/> <allow users="*/> </authorization> <location path="Premier"> <system.web> <authorization> <allow roles="Premier"/> </authorization> </system.web> </location>`
- D. `<authorization> <deny users=?"/> </authorization> <location path="Premier"> <system.web>`

**Answer: A**

### Explanation URL Authorization

Internet Information Services (IIS) 6.0 works with Authorization Manager, a management tool that is available with the Microsoft® Windows® Server 2003 family of operating systems, to implement IIS URL authorization. Overview Authorizing user access to Web application resources requires the management of many Access Control Lists (ACLs). In turn, maintaining ACLs requires administrators to track precisely which permissions are needed on each resource for each user or group to perform meaningful tasks. IIS URL authorization allows Windows administrators to simplify access management by authorizing user access to the URLs that comprise a Web application. When a user requests access to a URL, IIS URL authorization validates the user's access based on that

user's roles, which can be defined in Lightweight Directory Access Protocol (LDAP) queries, custom user roles, and Authorization Manager scripts (BizRules). This allows administrators to simplify access control management by controlling all user access to URLs instead of controlling access per ACL on each resource. IIS URL authorization is implemented as an Internet Server API (ISAPI) interceptor (in the diagram below, URL Authz ISAPI). When an application, virtual directory, or URL is configured to use IIS URL authorization, each request to a URL will be routed to the URL authorization ISAPI interceptor. The URL authorization ISAPI interceptor will use Authorization Manager (in the diagram, .NET Authz Framework) to authorize access to the requested URL. The URL must be associated with an Authorization Manager policy store that contains the authorization policy for the URL. Once the client has been authorized to access the URL, the URL authorization ISAPI's Execute URL feature (in the diagram, ExecURL) will pass the request to the appropriate handler for the URL, such as ASP.dll, another ISAPI, or the Static File Handler.



By using IIS 6.0 URL authorization, an administrator can control access based on information that is only available at runtime. For example, if you have a Web page that should only be available to employees in a given cost center or to employees of a certain age, you can assign roles to the correct users based on LDAP queries that will check the cost center or age attributes on a user's object. If employees can only access certain pages on certain days of the week or during a certain time of day, a BizRule can be created which grants access to the URL based on these values or any value that can be asserted at runtime, including IIS Server Variables. Using URL Authorization To use URL authorization in IIS 6.0 you must enable the ISAPI interceptor, Urlauth.dll. In addition, you must set the following metabase properties on the application, virtual directory, or URL (Web site):

- 1 **AzEnable:** Enables URL authorization for the virtual directory, application, or URL that corresponds to the entry in the metabase.
- 2 **AzStoreName:** Associates an Authorization Manager store with the virtual directory, application, or URL.
- 3 **AzScopeName:** Associates the virtual directory, application, or URL with a scope. This scope will be the name of a scope in the IIS 6.0 URL authorization application in the Authorization Manager policy store referred to in the AzStoreName attribute. If no scope or an empty string is specified, the default scope of the IIS 6.0 URL authorization will be used.
- 4 **AzImpersonationLevel:** Determines the impersonation behavior for the application. This allows you to configure the Web application to impersonate the client user, the IIS worker process, or the IUSER\_computername account for the worker process. Each setting significantly changes the environment and implied design of the Web application. Sample Script The sample script below, written in Microsoft Visual Basic® Scripting Edition (VBScript), marks the root of the first site as a URL in "MyAZScope", which is defined in the MyAZStore.xml file. Users with URLAccess rights in this scope will be able to access the site. `var objVDir = GetObject("IIS://localhost/w3svc/1/root"); objVDir.AzEnable = true; objVDir.AzStoreName = "MSXML://d:\MyAZStore.xml"; objVDir.AzScopeName = "MyAZScope"; objVDir.AzImpersonationLevel = 0; objVDir.SetInfo();` While URL authorization controls access to other forms of authorization, such as ACLs or IIS directory security permissions settings, the application context still requires the correct IIS directory security and ACL permissions. IIS URL authorization allows the IIS directory security and ACL permissions to be more easily maintained. When IIS 6.0 URL authorization is configured, the **AzStoreName** attribute in the IIS metabase entry for the application, virtual directory, or URL will identify an Authorization Manager policy store. To manage the authorization policy, run Authorization Manager and use the Open Policy Store. IIS 6.0 URL authorization is an

application in this store. The **AzScopeName** attribute in the metabase entry will be an authorization manager scope in the IIS 6.0 URL authorization application. Use this scope to manage access to the corresponding URL. When configuring an application, virtual directory, or URL for URL authorization, a scope must be created in the authorization policy store with the same name as that specified in the corresponding metabase entries **AzScopeName** attribute. Enabling the ISAPI Interceptor To use the URL authorization ISAPI interceptor (Urlauth.dll), you must first enable it for each Web site that requires URL authorization. **Important** You must be a member of the Administrators group on the local computer to perform the following procedure (or procedures), or you must have been delegated the appropriate authority. As a security best practice, log on to your computer using an account that is not

in the Administrators group, and then use the Run as command to run IIS Manager as an administrator. From the command prompt, type `runas /user:administrative_accountname "mmc %systemroot%\system32\inetmgr\iis.msc"`. To enable the URL authorization ISAPI interceptor

- 1 In IIS Manager, expand the local computer, expand the Web Sites folder, right-click the Website that you want, and then click Properties.
- 2 Click the Home Directory tab, and then in the Application settings section, click Configuration.
- 3 Click the Mappings tab, and then in the Wildcard application maps section, click Insert.
- 4 In the Add/Edit Application Extension Mapping box, click Browse and browse to the `Windows\system32\inetmgr` directory.
- 5 Click `urlauth.dll`, click Open, and then click OK. Related Topics

• For more information on Authorization Manager, see [Authorization Manager in Windows Help](#). The `<authorization>` Element Configures ASP.NET authorization support. The `<authorization>` tag helps control client access to URL resources. This element can be declared at any level (machine, site, application, subdirectory, or page). `<configuration> <system.web> <authorization> <authorization> <allow users="comma-separated list of users" roles="comma-separated list of roles" verbs="comma-separated list of verbs"/> <deny users="comma-separated list of users" roles="comma-separated list of roles" verbs="comma-separated list of verbs"/> </authorization>`

#### Subtags

| Subtag                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>&lt;allow&gt;</code> | Allows access to a resource based on the following: <b>users:</b> A comma-separated list of user names that are granted access to the resource. A question mark (?) allows anonymous users; an asterisk (*) allows all users. <b>roles:</b> A comma-separated list of roles that are granted access to the resource. <b>verbs:</b> A comma-separated list of HTTP transmission methods that are granted access to the resource. Verbs registered to ASP.NET are GET, HEAD, POST, and DEBUG.                                                  |
| <code>&lt;deny&gt;</code>  | Denies access to a resource based on the following: <b>users:</b> A comma-separated list of user names that are denied access to the resource. A question mark (?) indicates that anonymous users are denied access; an asterisk (*) indicates that all users are denied access. <b>roles:</b> A comma-separated list of roles that are denied access to the resource. <b>verbs:</b> A comma-separated list of HTTP transmission methods that are denied access to the resource. Verbs registered to ASP.NET are GET, HEAD, POST, and DEBUG. |

**Remarks** At run time, the authorization module iterates through the `<allow>` and `<deny>` tags until it finds the first access rule that fits a particular user. It then grants or denies access to a URL resource depending on whether the first access rule found is an `<allow>` or a `<deny>` rule. The default authorization rule in the `Machine.config` file is `<allow users="*" />` so, by default, access is allowed unless configured otherwise.



Top of page Example The following example allows access to all members of the Admins role and denies access to all users.

```
<configuration> <system.web><authorization><allow roles="Admins"/> <deny users="*/"/></authorization> </system.web></configuration>
```

### New Question Added

#### Question: 68

You are an application developer for your company. You create a Web application that is used by all users in the company. The application is hosted on the intranet Web server, which is named WebServer. WebServer has IIS 5.0 installed. The Web application is configured to use Integrated Windows authentication. The Web.config file specifies that the authentication mode is set to Windows. The application connects to a Microsoft SQL Server database named DataStore. The database is located on WebServer. The SQL Server computer is configured with SQL Server logins disabled. The database connection code is shown in the following code segment.

```
Dim myConnStr As String myConnStr = "Initial Catalog='DataStore';" myConnStr = myConnStr & "Data Source=localhost;Integrated Security=SSPI;" Dim myConn As New SqlConnection(myConnStr) Dim myInsert As String myInsert = "INSERT INTO Customer (CustomerID, Name) Values('123', 'John Doe')" Dim myCmd As New SqlCommand(myInsert) myCmd.Connection=myConn myConn.Open() myCmd.ExecuteNonQuery() myCmd.Connection.Close()
```

When you run the application by using Microsoft Internet Explorer, you receive an error message that reads in part: "Login failed for user WebServer\ASPNET." You need to ensure that the application can run successfully without prompting the user for a user name and password.

What should you do?

- A. Change the authentication mode in IIS to basic authentication. Update the connection string.
- B. Change the authentication mode in IIS to Anonymous and supply a login ID and password for a SQL Server login account that has access to the database. Update the connection string.
- C. Enable Integrated Windows authentication in Internet Explorer.
- D. Enable impersonation in the Web.config file.

**Answer: D**

#### Question: 69

4 You are an application developer for your company. You maintain a Windows Forms application. Data entry logic for the application is enforced by the user interface layer. The application contains assemblies that communicate data changes to the database. The application also contains assemblies that implement business logic. You create a new assembly named NewAssembly, which is called from the user interface. Values are passed to NewAssembly, which performs calculations by using the data. NewAssembly calls a separate assembly to store the resulting data in a database. You need to perform unit testing on the application to identify security vulnerabilities caused by unanticipated use of the application. Which two actions should you perform? (Each correct answer presents part of the solution. Choose two.)

- A. Test the application by calling NewAssembly directly.
- B. Test the application to verify whether it performs to the original functional specifications.
- C. Test the application by using a domain administrator account.
- D. Test the application by using the account of a user who should not have access to the application.

**Answer: A, D**

**Question: 70**

You are an application developer for your company. The company maintains an internal, selfsigned certification authority (CA). You are releasing a new internal Windows Forms application. A written company policy prohibits internal applications from running on client computers unless the identity and integrity of those applications can be proven. The Microsoft .NET Framework on all client computers is configured to enforce this restriction. You need to ensure that your application will run when installed on all client computers. Your solution must not require any financial expenditure. What should you do?

- A. Use a software publisher certificate issued by the internal CA to sign the application assemblies.
- B. Use a software publisher certificate issued by a third-party commercial CA to sign the application assemblies.
- C. Run the Certificate Creation tool and the Software Publisher Certificate Test tool before distributing the application to client computers.
- D. Distribute the application as an e-mail attachment. Digitally sign the e-mail message before sending it to all company users.

**Answer: A**

**Question: 71**

You are an application developer for your company. You create an ASP.NET Web application. The application allows customers to select items for purchase. During the active session of a customer, data about the quantity and price of items selected by the customer is stored in a cookie on the client computer. You need to test the application for security vulnerabilities. What should you do?

- A. Test the application by using a browser that has cookies disabled.
- B. Test the application by selecting 150 items for purchase.
- C. Test the application by using a cookie that you create in a text editor.
- D. Test the application by using the five most common Internet browsers.

**Answer: C**

**Question: 72**

You are an application developer for your company, which is named Humongous Insurance. You are developing an application to manage medical insurance claims. The application includes a serviced component named ClaimRecord. The business rules implemented by the application allow only those users who are members of the HumongousInsurance\ClaimsProcessor domain group to access the ClaimRecord component. You apply attributes to the ClaimRecord component to enable role-based security. You use the following assembly-level attribute to add a role named ClaimsProcessor to the COM+ application that hosts the ClaimRecord component. <Assembly: SecurityRole("ClaimsProcessor")> You deploy the ClaimRecord component to your staging server. You log on to the application by using a user account that is a member of the HumongousInsurance\ClaimsProcessor domain group. When your application attempts to access the ClaimRecord component, an UnauthorizedAccessException exception is thrown. You need to modify the ClaimRecord component or reconfigure the COM+ application so that access is granted. You need to achieve this goal without compromising the security requirement of the ClaimRecord component. What should you do?

- A. Replace the assembly-level attribute with the following attribute. <Assembly: SecurityRole("ClaimsProcessor", SetEveryoneAccess:=True)>
- B. Replace the assembly-level attribute with the following attribute. <Assembly: SecurityRole("HumongousInsurance\ClaimsProcessor")>
- C. Add the SuppressUnmanagedCodeSecurity attribute to the ClaimRecord component.
- D. Using the Component Services tool, add the HumongousInsurance\ClaimsProcessor domain group to the COM+ ClaimsProcessor role.

**Answer: D**

**Question: 73**

You are an application developer for your company. The company runs an e-commerce Web site. Users log on to the Web site by using a password. Passwords are stored in a text file. The following code segment prepares the passwords for storage. Function HashPassword(ByVal Pwd As String) As String Return FormsAuthentication.HashPasswordForStoringInConfigFile(Pwd, "SHA1") End Function Users of the Web site are creating passwords that are easily cracked by dictionary attacks. You need to decrease the likelihood that a dictionary attack will succeed if the password file is stolen, without restricting the passwords that users can create. What should you do?

- A. Create a dictionary file that contains common words. Write additional code to reject passwords that match the entries in the dictionary.
- B. Apply a more restrictive discretionary access control list (DACL) to the password storage file.
- C. Replace the HashPassword function with the following code segment. Function HashPassword(ByVal Pwd As String) As String Return FormsAuthentication.HashPasswordForStoringInConfigFile(Pwd, "MD5") End Function
- D. Replace the HashPassword function with the following code segment. Function HashPassword(ByVal Pwd As String) As String Dim Rng As New RNGCryptoServiceProvider Dim Salt(16) As Byte Rng.GetBytes(Salt) Dim saltstr As String = Convert.ToBase64String(Salt) Return saltstr & FormsAuthentication.HashPasswordForStoringInConfigFile( \_ saltstr & Pwd, "SHA1") End Function E. Replace the HashPassword function with the following code segment. Function HashPassword(ByVal Pwd As String) As String Dim Hash As Integer = 0 Dim Enc As New UnicodeEncoding Dim HashData As Byte() = Enc.GetBytes(Pwd) Dim i As Integer For i = 0 To HashData.Length Step 2 Hash = Hash Xor (HashData(i) Or (HashData(i + 1) << 8)) Next Return Hash.ToString() End Function

**Answer: D**

**Question: 74**

You are an application developer for your company. You develop an ASP.NET Web application for the company's intranet. The application accesses data that is stored in a Microsoft SQL Server database. Access to objects in the database is granted based on the identity of the user of the application. The application uses Windows authentication, and it has impersonation enabled. You need to modify the application so that it also uses a new serviced component. The new component requires applications that call it to have membership in the COM+ role named AuthorizedCallers. The developer who developed the new component creates a new Windows user account named InternalWebAppUser and adds this user account to the COM+ AuthorizedCallers role. The developer instructs you to write your application to access the serviced component by using the security context of this user account. You need to modify your code to call the new serviced component by using the security context of the InternalWebAppUser user account. What should you do?

- A. Disable impersonation in the Web.config file and configure the ASP.NET worker process to run by using the InternalWebAppUser user account.
- B. Set the authentication mode to None in the Web.config file.
- C. Modify the database connection string to connect as the InternalWebAppUser user account.
- D. Write code to impersonate the InternalWebAppUser user account for each call to the serviced component.

**Answer: D**

**Question: 75**

You are an application developer for your company. You are developing an ASP.NET Web application that will use Forms authentication to authenticate each user who attempts to use the application. The application will store user names and passwords in the <credentials> section of the Web.config file. You need to configure and implement Forms authentication for the application. Which four actions should you perform? (Each correct answer presents part of the solution. Choose four.)

- A. Add the following element to the Web.config file. <authentication mode="Forms"> <forms loginUrl="logon.aspx" name="myAuthCookie" path="/" /> </forms></authentication>
- B. Add the following element to the Web.config file. <authorization> <deny users="?" /> <allow users="\*" /></authorization>
- C. Create a Logon.aspx page that includes a Logon button. Add a Click event handler for the Logon button.

- D. Add an Application\_OnAuthenticate event handler.
- E. Add code that calls FormsAuthentication.Authenticate to authenticate a user.
- F. Add code that creates an IPPrincipal object and associates it with the current HTTP context.

**Answer: A, B, C, E**

**Question: 76**

You are an application developer for your company. You develop a library assembly that contains diagnostic utility classes. This library assembly is installed in the global assembly cache on all client computers on the company network. You develop a Windows Forms application that calls the library assembly. You successfully test the application on your computer, and then you deploy the application to a Web folder on the intranet. Further testing reveals that when you run this application from the intranet, a SecurityException exception is thrown when the application is loading. You need to correct the problem that is causing the SecurityException exception. What should you do?

- A. Add the following code segment to the library assembly. <Assembly: AllowPartiallyTrustedCallers()>
- B. Add the following code segment to the Windows Forms application assembly. <Assembly: AllowPartiallyTrustedCallers()>
- C. Add the following code segment to the library assembly. <Assembly:
   
PermissionSet(SecurityAction.RequestOptional, Name:="LocalIntranet")>
- D. Add the following code segment to the Windows Forms application assembly. <Assembly:
   
PermissionSet(SecurityAction.RequestMinimum, Name:="LocalIntranet")>

**Answer: A**

**Question: 77**

You are an application developer for your company. You are developing an application that calls a Web service on the company intranet. This Web service uses Integrated Windows authentication to authenticate callers. The application must be authenticated by the Web service.

You plan to use a proxy class named ProductInfo to invoke a Web service method named GetProductCount. You write the following code segment. (Line numbers are included for reference only.)

```
1 Public Function GetInventory(ByVal name As String) As Integer
2 Dim productProxy As ProductInfo = New ProductInfo
3 '...
4 Return productProxy.GetProductCount(name)
5 End Function
```

You need to set the credentials that will be used by the call to the Web service method. You need to add code at line 3 of the code segment to achieve this goal.

Which code segment should you use?

- A. Dim cred As NetworkCredential = New NetworkCredential(\_userName, \_psswd, \_domain)Dim cache As CredentialCache = New CredentialCachecache.Add(New Uri(productProxy.Url), "Basic", cred)productProxy.Credentials = cache
- B. Dim cred As NetworkCredential = New NetworkCredential(\_userName, \_psswd, \_domain)Dim cache As CredentialCache = New CredentialCachecache.Add(New Uri(productProxy.Url), "Windows", cred)productProxy.Credentials = cache
- C. productProxy.Credentials = CredentialCache.DefaultCredentials
- D. Dim identity As GenericIdentity = New GenericIdentity(\_userName, "Windows")Dim principal As GenericPrincipal = New GenericPrincipal(identity, Nothing)Thread.CurrentPrincipal = principal

**Answer: C**

**Question: 78**

You are an application developer for your company. You are conducting a code review of a Windows Forms application that was developed by another developer. The application contains code that validates a user's ability to access restricted functionality. A

variable named `bolElevated` contains a value of `True` when a user logs on by using a user account named `Admin`. Otherwise, the variable contains a value of `False`. If a user other than `Admin` attempts to access the restricted functionality, the application must display an error message. A function named `OpenAdmin()` displays the user interface for the restricted functionality. The application contains the following code segment. If `bolElevated` Then `OpenAdmin()` Else `MsgBox "That functionality is restricted." MsgBox "To access restricted functionality, log in as user 'Admin'."` End If You need to improve the security of this code segment while maintaining its functionality. You decide to replace the existing code segment.

Which code segment should you use?

- A. If `bolElevated` Then `OpenAdmin()`Else `MsgBox "That functionality is restricted."`End If
- B. If `bolElevated` Then `OpenAdmin()`Else `Console.WriteLine "That functionality is restricted."`End If
- C. If `bolElevated = False` Then `OpenAdmin()`Else `MsgBox "That functionality is restricted." MsgBox "To access restricted functionality, log in as user 'Admin'."`End If
- D. If `bolElevated = True` Then `OpenAdmin()`Else `MsgBox "That functionality is restricted." MsgBox "To access restricted functionality, log in as user 'Admin'."`End If

**Answer: A**

**Question: 79**

You are an application developer for your company. You are developing an application that stores and retrieves data in a Microsoft SQL Server database. Part of the application accepts a value from a third-party component and stores the value in a variable named `intInput`. The application then inserts the value of `intInput` into a `tinyint` column in a SQL Server table. You need to improve the security of the application when the contents of `intInput` are used in a SQL query. What should you do?

- A. Convert the contents of `intInput` to a string value.
- B. Ensure that the contents of `intInput` are compatible with the SQL Server `tinyint` data type.
- C. Ensure that the contents of `intInput` do not include any Base-64 encoding.
- D. Ensure that the contents of `intInput` do not include any single or double quotation marks.

**Answer: B**

**Question: 80**

You are an application developer for your company. You are developing an ASP.NET Web application. All users in the company use Microsoft Internet Explorer 6.0. A group of users is testing the application. The users report that when an exception occurs, the full exception information is displayed in their Web browsers. You need to ensure that the full exception information is not displayed when an exception occurs. What should you do?

- A. Require users to use HTTPS to access the application.
- B. Trap all exceptions and display a generic error message.
- C. Instruct users to enable friendly error messages in Internet Explorer.
- D. Obfuscate the compiled assemblies of the application.
- E. Modify the application's configuration to disable custom errors.

**Answer: B**

**Question: 81**

You are an application developer for your company. You are developing a Windows-based payroll application that will be used by all payroll administrators in the company. The application has a single executable file that uses a separate assembly to modify payroll data. You need to design security for your application to ensure that the assembly cannot be called by unauthenticated and unauthorized users. What should you do?

- A. Run the application by using a user account that has access to the application directory.

- B. Modify the application to validate all user-entered data.
- C. Modify the application to authenticate and authorize user access within each assembly as it is called.
- D. Modify the application to authenticate and authorize user access when each user runs the executable file.
- E. Set the folder-level permissions to the executable file by using directory security.

**Answer: C**

**Question: 82**

You are an application developer for your company. You are conducting a code review of an ASP.NET Web application that uses Forms authentication. The application uses the following code segment to create a new user account and add the account to a database by using a custom class named FormUser.

```
Public Class FormUser <Flags()> _ Enum UserPerms AccessPayroll = 1 AccessHRData = 2 AccessBlueprints = 4 End Enum Dim
Perms As UserPerms Private Sub RemovePermission(ByVal Gone As UserPerms) Perms = Perms And Not Gone End Sub Private
Sub AddPermission(ByVal Gone As UserPerms) Perms = Perms Or Gone End Sub Public Sub SetPerms(ByVal Department As
String) Perms = UserPerms.AccessPayroll Or UserPerms.AccessHRData _ Or UserPerms.AccessBlueprints Select Case
Department Case "Engineering" RemovePermission(UserPerms.AccessPayroll Or UserPerms.AccessHRData) Case "Accounting"
RemovePermission(UserPerms.AccessHRData Or UserPerms.AccessBlueprints) End Select End Sub End Class
```

You need to improve the security of this code. Which two actions should you perform? (Each correct answer presents part of the solution. Choose two.)

- A. Add the following code segment to the case statement. Case Else Throw New ApplicationException("Unknown department")
- B. Change the UserPerms enumeration as follows. Enum UserPerms AccessPayroll AccessHRData AccessBlueprintsEnd Enum
- C. Change the initial part of the SetPerms subroutine as follows. Perms = 0Select Case Department Case "Engineering" AddPermission(UserPerms.AccessBlueprints) Case "Accounting" AddPermission(UserPerms.AccessPayroll)
- D. Replace the code segment with the following code segment. Public Class FormUser <Flags()> \_ Enum UserPerms ProhibitPayroll = 1 ProhibitHRData = 2 ProhibitBlueprints = 4 End Enum Dim Perms As UserPerms Private Sub AddProhibition(ByVal Add As UserPerms) Perms = Perms Or Add End Sub Public Sub SetPerms(ByVal Department As String) Perms = 0 Select Case Department Case "Engineering" AddProhibition(UserPerms.ProhibitPayroll Or UserPerms.ProhibitHRData) Case "Accounting" AddProhibition(UserPerms.ProhibitHRData Or UserPerms.ProhibitBlueprints) End Select End SubEnd Class

**Answer: A, C**

**Question: 83**

You are an application developer for your company. You create a Windows Service application that runs under the local System account on each user's client computer. This application creates a new folder for each exception event that occurs. The application design specifies that these new folders do not inherit the permissions of the parent folder. The application adds text files to the new folders for troubleshooting. These files must be accessible only by the local System account and by specific members of the support staff. After you deploy the application, the specific members of the support staff report that they cannot access the files. You need to ensure that the specific members of the support staff can access the files. What should you do?

- A. Ask the network administrator to grant the Everyone account discretionary access control list (DACL) permissions for the folders and files.
- B. Ask the network administrator to remove all permissions from the folders that do not belong to the System account and set the parent folder of the new folders to allow inheritable permissions to propagate from the parent folder to the new folders.
- C. Create an assembly that the Windows Service application calls to set the discretionary access control list (DACL) permissions for the folders and files.
- D. Create an assembly that the Windows Service application calls to impersonate the System account that grants permissions for the folders and files to the System account.

**Answer: C**

**Question: 84**

You are an application developer for your company. You develop a Windows Forms application. You set a breakpoint in the code and run the application. You receive an error message stating that you are unable to start debugging. You need to be able to debug the application without having more authority than necessary. What should you do?

- A. Add your user account to the local Debugger Users group.
- B. Grant your user account complete control of the executable file for the application.
- C. Modify the code access security policy to run the Microsoft Visual Studio .NET debugger with the FullTrust permission.
- D. Run Microsoft Visual Studio .NET 2003 by using the runas command and specify a user account in the Power Users group.

**Answer: B**

**Question: 85**

You are an application developer for your company. You are reviewing the security for a console application that was written by another developer. The application uses impersonation to run as a member of the Administrators group. The following code segment is the only code that deals with security in the application. Dim key As RegistryKey key = Registry.CurrentUser.CreateSubKey("Name") key.SetValue("Name", "Tester") You need to improve the security of the application. What should you do?

- A. Change the application to run as the interactive user.
- B. Run the application from the command line by using the runas command and specify the Administrator account.
- C. Change the application to use code access security.
- D. Change the application to write to the HKEY\_LOCAL\_MACHINE hive.

**Answer: A**