

**Exam 70-330 study material**

**Made available by CertsKing.com**



**Free 70-330 Exam Preparation Questions**

**Exam 70-330: Implementing Security for Applications with Microsoft Visual Basic .NET**

**Question: 1**

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 examining 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 = "Local")]
- 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 an 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 state 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 an Assert you will make luring attacks possible.

**Question: 2**

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 by 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

- ```

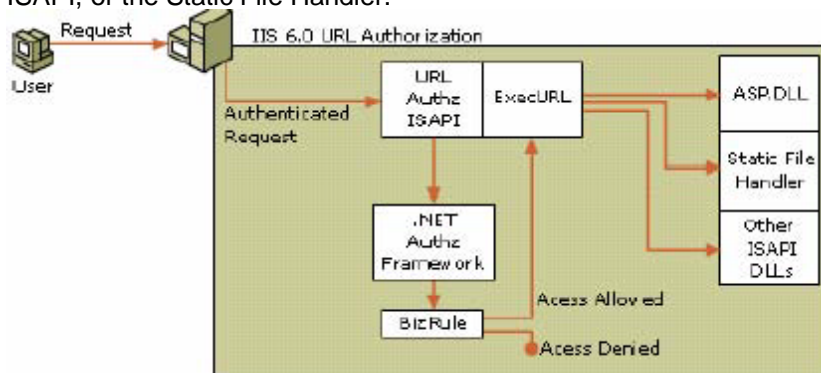
roles="Premier"/> <deny users="*" />
</authorization>
</system.web>
</location>

```
- 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 t 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. hen 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<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 |
|--------|-------------|
|--------|-------------|

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <allow> | 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.                                                  |
| <deny>  | 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: 3

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: 4**

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: 5**

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: 6**

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: 7**

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: 8**

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: 9**

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: 10**

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**

For complete [Exam 70-330 Training kits and Self-Paced Study Material](http://www.certsking.com/70-330)

Visit:

<http://www.certsking.com/70-330.html>

**CERTSKING**

The Best Leader In Certifications

<http://www.certsking.com/>

