Global Information Assurance Certification Paper

Creating a Secure n-tier DCOM Implementation

Pranav Jhumkhawala

July 23, 2002

GSEC v1.4, option 1

Abstract

While DCOM offers the capability of developing n-tier applications that are flexible and scalable, it also poses several challenges to the developer, security being the most significant, and often the overlooked one. By using the underlying security features of DCOM and other technologies such as firewall and Windows security, it is possible to have a secure and manageable DCOM implementation that serves the business and customer needs.

This paper provides an overview of DCOM security principles and discusses the security considerations that need to be taken into account while implementing a typical DCOM installation. Various components that make up the multi-tier DCOM implementation, such as MTS, Firewall configuration and so on, are discussed at an appropriate level of detail. Where applicable, the paper makes recommendations regarding what settings should be used in order to have a secure DCOM implementation.

Creating a secure n-tier DCOM Implementation

1 Introduction

Present day business applications, especially web applications, are designed as multi-tier (n-tier) client server applications. This type of architecture is popular because it allows true separation of data layer, business logic layer and presentation layer, making each of those layers more manageable. Popularity of this type of architecture gave rise to software systems based on distributed objects, such as Common Object Request Broker Architecture (CORBA), Remote Method Invocation (RMI) and Distributed Component Object Model (DCOM). These technologies have grown quite significantly in terms of functionality and efficiency over the last few years. Application server technologies such as Enterprise Java Beans (EJB) and Microsoft Transaction Server (MTS) are becoming more and more popular. These application server technologies provide features such as component pooling, database connection management, object persistence etc.

Security is an important aspect that needs to be taken into consideration in any architecture. The issue is more significant in case of architecture described above because the components that are hosted by these application servers typically contain sensitive business logic and proper definition of who can invoke the components and where and how can the information travel is essential.

The technologies mentioned above provide, in their own peculiar way, tools and configuration options to secure the components that are hosted by these application servers and the underlying distributed object technology provides ways of securing the communication between application servers, database servers and web servers.

DCOM model is developed by Microsoft. Support for Distributed COM was made available with first release of Windows NT 4, around August 1995. DCOM is built based on Remote Procedure Call mechanism, which is an industry standard. DCOM has several requirements, which are not present when one looks at just the COM compliant objects. One of the most significant one is, of course, security. In order to satisfy this requirement Windows provides a robust security layer that integrates seamlessly with COM and allows authentication and authorization of accesses to remote server processes.

2 Understanding DCOM Security Policies

DCOM has an extensible and customizable security framework built in which allows the deployment of security. DCOM security addresses four different aspects of security

Access security: Who can call an object?
This caters to the need of distributed application to protect objects against unauthorized access. Depending on the business requirements, the components can be accessed by only predefined users, or can be invoked by any user, but may not be able to access all the features. The component security can be configured to allow appropriate access at deployment time. Additionally, security can be implemented programmatically (within component itself) to grant or deny access at component or even at individual method level.

Launch security: Who can create a new object in a new process?
Once a user has access to the server hosting the COM components, that user can potentially invoke any component on that server. Therefore it is important to secure the components so that unauthorized users cannot create components they are not supposed to. By very nature of this security issue, this has to be handled not programmatically, as the fact that someone can execute the program is a security breach in itself, which could be exploited to launch a denial of service attack. The COM libraries perform a special validation when the object create request is received. The privilege information stored in the registry about the object is utilized to determine if the launch request is authorized.

Identity: Who is the owner of the object itself?
In COM, the object performs actions on behalf of the caller. In other words the object assumes the identity of the caller. Based on the privileges of the caller, the object may or may not be able to perform certain function. This works well in a situation where there's only one user accessing the component. However, a more realistic scenario is where a number of users who may invoke that component. In this case, the objects and users as well as the resources accessed by the object will need to be configured with same privileges. The problem is that if these are too restrictive, then users may not be able to access all features uniformly, and if it is to permissive, it can pose security threat. Additionally Internet applications do not assign a dedicated user account for every user, therefore identifying users may not be possible. The identity security setting can be set to one of the three listed below in DCOM.

Run as activator – The component is created using the identity of the caller. This is the default setting. Disadvantage is that if different clients invoke the same object on the server, different processes will be created for each client, which could have adverse effect on the component's performance.

Run as interactive user – Component is created using the identity of currently logged on user on the server. This means that for object to be created successfully, a user with proper authority will have to be logged on the server otherwise the creation will fail.

Run as "This" user – A user account is specified under whose identity the components will be created. This account can be a non-interactive account. Typically this account is part of MTS Trusted Impersonator group.

Connection policy:

This can be further classified in three categories –

1. Integrity - can someone alter the communicated message?
2. Privacy – are others allowed to see the communication besides the caller?
3. Authentication – how much can the object trust the caller?

3 DCOM Security Policies Implementation

In DCOM, authentication, data integrity and privacy are all wrapped up in a single property called the *Authentication Level*. The authentication levels can be set on a per-application (component) basis or they can be set as a default setting for the server, which will be inherited by all the components hosted on that server.

There are seven authentication levels of DCOM. Security will increase as the level increases, however, there's added performance cost that may need to be considered. Additionally, since the authentication levels are set in the hierarchy where higher levels are based upon the lower levels, the strength as well as weaknesses of the lower levels are inherited in the higher ones. In other words, if say the Level 2 authentication has certain vulnerability, setting authentication level to Level 7 does not necessarily get rid of the said vulnerability.

The DCOM authentication levels are –

Level 1 – No Authentication: Least secure. Allows client to connect and invoke components on the server without requiring validating the remote application.

Level 2- Connect Authentication: Client is validated when the connection is made to the server. If using Windows 2000, and if the client machines are also Windows 2000 servers, Kerberos can be used as the authentication methods. Otherwise in an NT environment authentication will be done using the NTLM protocol. Note that in this level of authentication, subsequent packets are not authenticated.

Level 3 – Default Authentication: Authentication is done using the underlying security architecture of the operating system. This is same as Connect Authentication, since in Windows, currently there is only one security architecture.

Level 4 – Call-level Authentication: Authentication is made every time the client invokes a method in the server object. It is important to remember here that

authentication of each method call by the client does not necessary require the client to enter the user id and password for each of those calls. The remote machine can cache the password. Calls are broken into multiple packets and only one of the packets contains the authentication information. Due to this, during a hijacking attack, the rest of the packets can still be replaced with malicious information. Even the packet containing the authentication information can also be modified or replaced.

Level 5 – Packet-level Authentication: Each packet will be authenticated. While it fixes the problem of call-level authentication wherein packets not containing authentication information can be replaced or modified, it still does not prevent the packets with authentication information from being modified or replaced.

Level 6 – Packet Integrity-level Authentication: In this level, a checksum is added to each packet to ensure integrity of the packets and prevent modification or replacing of packets. Sophisticated user can still read the data flowing through the network. If information being passed is highly sensitive, this authentication level may not be sufficient.

Level 7 – Packet Privacy-level Authentication: Builds on the previous level of authentication by encrypting all data contained in packets. The data is protected and packet tempering can be prevented as long as the encryption algorithm being used is robust.

4 Using DCOM Configuration utility (DCOMCNFG.EXE)

The DCOM Configuration utility allows users to set many of the security settings of the DCOM applications on the server.

The utility consists of three main tabs –

1. Applications
2. Default Properties
3. Default Security

Applications Tab

The Applications tab lists all the objects that are available on this machine and can be launched. More details of each of these objects are available by selecting the Properties option. Here one can verify which server (remote or local) the object is hosted as well as identity of the user under which this component will run. Following settings are available in the properties option for each object:

General – provides general information about the object.

Location – determines where the DCOM object will be executed.
Security – here, user can configure access, launch and configure permissions for the selected object. Note that these settings will be specific to each object. If no settings are specified here, the default settings as described under explanation for other tabs will come into effect.
Identity – specifies account that will be used to launch this object. There are four choices available –

1. The Interactive user – that is the current logged on user
2. The Launching user – user account that initiated the object.
3. This user – specify a user account that will be always used to launch the object.
4. The System account – object will be launched using security context of the System account.

Default Properties Tab

In this tab, the global settings for entire machine with respect DCOM configuration and behavior are managed. Users can enable or disable DCOM on the machine by checking or clearing the "Enable Distributed COM on This Computer" option.

It is also here that a user will set the Default Authentication Level for DCOM communications. The authentication levels of DCOM are explained in the previous section. The default authentication level is "Level 5 – Packet-level Authentication".

Default Security Tab

There are three options under Default Security Tab –

1. *Default Access Permission* - This value determines the users and groups that can access an object when no other access permissions are provided.
2. *Default Launch Permission* - This value determines the users and groups that can launch an object when no other access permissions are provided.
3. *Default Configuration Permission* - This value determines the users and groups that may read or modify configuration information for DCOM applications. This also includes which users and groups will have permission to install new DCOM servers.

5 Configuring and implementing remote objects

With the background of how DCOM security works as discussed above, let's now look into the configuration and implementation of remote objects that use DCOM technology. The COM components can be developed using Microsoft Visual C++

and Microsoft Visual Basic. Other Microsoft development languages such as Microsoft Visual J++ as well as C-Sharp can also be used to develop COM components. Of these, Visual Basic is arguably the easiest language to develop COM components.

DCOM deployment architecture can vary significantly from one installation to another based on organization's needs and requirements for business rules as well as security policies. The following architecture is selected for the purpose of this discussion because, while it is no way a standard implementation, it covers components that allow discussion of majority of the issues surrounding DCOM security implementation. The architecture for the discussion is as follows:

- We have One (or more than one) COM DLLs developed in Visual Basic.
- These DLL(s) are deployed within one package in Microsoft Transaction Server (MTS).
- The MTS server is located inside the firewall and outside users have no access to this server.
- The client is a Web server located in the DMZ running IIS.
- The package is remotely deployed on the DMZ Web server.
- The communication between the Web server and the MTS server is performed through the firewall.
- All the servers are Windows NT 4.0 servers and they are set up as stand alone servers.

5.1 Configuring MTS Package

First of all, we need to enable authorization checking for the package.

1. Select the package you wish to configure.
2. On the *Action* menu, click *Properties* and select *Security* tab.
3. Check the *Enable Authorization Checking* check box. This is the default setting for MTS Packages.

Now, configure this package to run under identity of a specific user. If you have not already done so, create a user account and make that user account part of MTS Trusted Impersonator group. This is the user account under whose identity the components will be run. The same user id/password combination will need to be created on the client machine as well. This is necessary because in our example, we have a peer-to-peer relationship between the client and the server and no Domain authentication is available. The other option to make this work is to turn off the DCOM authentication, but by doing this, security is compromised.

Once you have created this account, you can set the MTS package to run under the identity of this user.

1. Select the package whose identity you wish to change.
2. On the *Action* menu, click *Properties* and select *Identity* tab.
3. Select the *This user* option and enter the user domain followed by a backslash (\), user name (one you created for this purpose), and password for the Windows NT user account.

After performing the above tasks, stop the package server processes by selecting the package, right clicking, and selecting Shut Down option. You can also shut down all packages at one time by selecting My Computer and choosing Shut Down Server Processes option from the Action menu.

Next step is to export the package to create client executables that will be run on the client machine to install the remote components on the client.

1. Select the package you want to export in the left pane of the Explorer.
2. On the *Action* menu, click *Export*. You can also right-click and select the *Export* option.
3. In the *Export Package* dialog box, enter or browse for the package file to create. The component files will be copied to the same directory as the package file.
4. If you want to include any roles that you have identified for the package, click the *Save Windows NT user ids associated with roles* checkbox.
5. Click *Export*.

The destination folder that you selected will contain a package file (with the .pak extension) containing information about the components and roles (if any) included in the original package, and copies the associated component files to the same directory in which the package file was created. Only component DLLs are copied. Package locks against changes or deleting will be exported with the package.

5.2 Configuring the client

As mentioned earlier, we will need to create a user account on the client machine that is defined by the same user id and password as the account under whose identity the components are configured to run. Create this account as appropriate and make it part of MTS Trusted Impersonator group.

Run the client executable file created by exporting package on the MTS server machine. This will create necessary registry entries on the client machine for the components. These settings can be verified using DCOM Configuration utility provided with Windows. We will discuss later how to use this utility to configure security settings.

5.3 Configuring DCOM to work across firewall

DCOM, while communicating across firewall, assigns one TCP port and one UDP port at run time to each process serving DCOM objects. While this feature of DCOM has it's own advantages and benefits, one disadvantage is that it is difficult to configure DCOM with firewall since at the run time, DCOM could potentially use any port between port number 1024 to port number 65535. Of course, it is possible to configure firewall to allow DCOM to communicate for this range of ports, but it creates a security hole. Therefore, on the server hosting the COM components certain settings must be done to restrict this range of firewall ports. But before that, we must configure DCOM to use TCP only, to avoid exposing firewall to unnecessary security risks that are coupled with using UDP. To do this, using regedt32.exe, move "NCACN_IP_TCP" value to the top of the list in the DCOM Protocols named value of the *HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc* registry key. A side-note not related to security here is that the same change should be made on the client machine as well in order to avoid 30-45 seconds delay when connecting to TCP only servers.

Restricting Range of TCP Ports

These settings must be done on server machines only. Using regedt32.exe, create the *HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Internet* registry key. Add following key values to the newly added key.

Name	Type	Value
Ports	REG_MULTI_SZ	Specify one port range per line. For example: 1500-2000 5141
PortsInternetAvailable	REG_SZ	Y (always)
UseInternetPorts	REG_SZ	Y

Value of "Y" in the UseInternetPorts key indicates that the DCOM application will use the ports listed under Ports key. The value of "N" in that key indicates that the DCOM application will not use the ports listed under Ports key. For additional security, set the UseInternetPorts value to "N" and make the DCOM application ask for permission to be accessible through the port range specified programmatically.

Configuring Firewall

The firewall should be configured as follows:

1. Deny all incoming traffic to the server.

2. Allow incoming traffic on TCP port 135 between the DMZ Client machine and the MTS server machine.
3. Allow incoming traffic on TCP port range specified in the registry settings as described above between the DMZ client machine and the MTS server machine.

6 Create a Trust Relationship

In our setup architecture we need to establish authentication between two stand-alone servers. Since these two servers are stand-alone, they are part of two different domains. Therefore even though we created same user account on both the client machine and the server machine, DCOM still cannot authenticate the user. This is because when the client machine passes the user's identity to the server, the identity is client_machine_name\user_name. On the server side, however, the identity is server_machine_name\user_name. Due to this, the two will not match and the authentication will fail. In order to alleviate this problem, we must setup a Trust relationship between the client machine and the server machine.

Trust relationship is a link between two different domains, where one domain trusts the users from the other domain. To create the trusted relationship –

1. Make sure you are logged onto the server machine.
2. From User Manager for Domains, choose Select Domain from the User menu. Type "Client Machine".
3. From the Policy menu, choose Trust Relationship. Choose Add and type "Server Machine". Enter a password that you will use on Server Machine to trust Client Machine. Server Machine should now be listed under "Permitted to Trust" this Domain. Close the Trust Relationship dialog box.
4. From the User menu choose Select Domain and type "Server Machine".
5. From the Policy menu, choose Trust Relationship. Add "Client Machine" and use the same password you used in Step 4.

A dialog box appears notifying you, "Trust relationship with Client Machine successfully established."

7 Conclusion

Microsoft's DCOM/COM+ architecture allows implementation of secure n-tier architecture. The architecture also as significant amount of flexibility built in. Most of the security features can be implemented either by using utilities such as Microsoft Management Control or DCOM Configuration utility, or it can be implemented programmatically by using available security APIs. The DCOM/COM+ architecture integrates seamlessly with the windows security and

provides ways to integrate with other security components such as firewalls, cross-domain authentication etc.

Having said that, being transparent and flexible to the developers, DCOM/COM+ security is often overlooked or poorly implemented. Due consideration must be given to the security aspects when designing the n-tier DCOM application. Depending on the business needs, appropriate authentication level should be set and firewall rules should be defined. It is also important to execute a thorough system performance testing since security implementation definitely have an impact on the performance of the system.

8 References

Microsoft Systems Journal. "The COM+ Security Model Gets You out of the Security Programming Business." November 1999. URL: http://www.microsoft.com/msj/defaultframe.asp?page=/msj/1199/comsecurity/comsecurity.htm&nav=/msj/1199/newnav.htm (September 2, 2002)

Shohoud, Yasser. "Programming COM+ Security." May 2000. URL: http://www.devx.com/upload/free/features/vcdj/2000/05may00/mt0500/mt0500.asp (September 2, 2002)

IT World.com. "COM+ Security Programming, Part 1: Declarative Role-Based Security." June 2001. URL: http://www.itworld.com/nl/windows_sec/06112001/ (September 2, 2002)

IT World.com. "COM+ Security Programming, Part 2: Programmatic Role-Based Security." June 2001. URL: http://www.itworld.com/nl/windows_sec/06182001/ (September 2, 2002)

Nelson, Michael. "Using Distributed COM With Firewalls." March 1999. URL: http://www.microsoft.com/com/wpaper/dcomfw.asp (September 2, 2002)

Microsoft Corporation. "Q176799 INFO: Using DCOM Config (DCOMCNFG.EXE) on Windows NT." January 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;[LN];Q176799 (September 2, 2002)

Pattison, Ted. "Programming Distributed Applications with COM and Microsoft Visual Basic 6.0." Microsoft Press, 1998.

Viega, John; McGraw, Gary. "Building Secure Software." Addison Wesley, 2002.

Hostman, Marcus; Kirtland, Mary “DCOM Architecture.” July 1997. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomarch.asp (September 2, 2002)

Microsoft Corporation. “Enabling MTS Package Security.” August 2002. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mts/install_9kvt.asp (September 2, 2002)