Global Information Assurance Certification Paper

Using and Securing DCOM

by

Brent Roskos

For

GIAC Security Essentials Certification (GSEC)
Version 1.4b, Option 1

August 17, 2003

Using and Securing DCOM

Abstract/Summary

Distributed computing is a big part of today's computing environment. DCOM is one method for extending computing resources between computers. DCOM is built-in to all Microsoft operating systems since Windows 98 and is available as an add-in for Microsoft Windows 95. The basic idea behind DCOM is to provide a standard environment with which developers can create distributed applications.

This paper will take a look at the information surrounding DCOM, and provide a brief summary of what is, how to use it or, and related security concerns. There is a great deal of information on the Internet regarding DCOM. Most of this information seems to be geared toward a software developer. In this paper, I will try it to put this information in a format where a system administrator or a security administrator can get the information that they need quickly and easily and come to a basic understanding of how the technology works and what can be done to secure it.

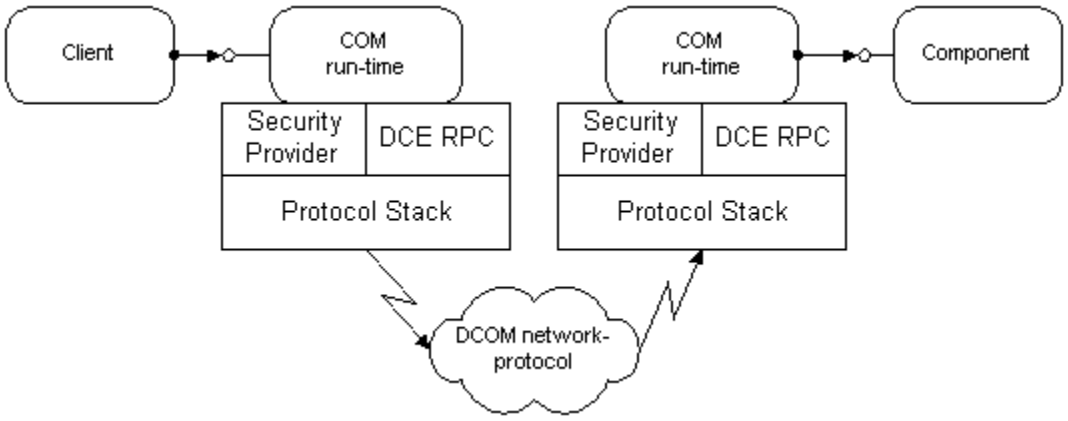
What is DCOM?

Microsoft distributed COM (DCOM) is built to extend to the component object model (COM) computing environment to function between multiple computers over a network. COM allows components on a single computer to interact with different components on the same computer. The operating system handles the communication between these components with IPC or inter-process communication. DCOM extends this functionality to work between computers; a component on one computer can take advantage of resources or data available to another component on a different computer. The communication between components over the network is handled by the operating system, this time using Remote Procedure Calls (RPC).

The functionality for providing security to allow or disallow the use of individual components either on the same computer or components on a different computer is built into the operating system and a standard for all applications using DCOM. This tool is launched from the command line by entering DCOMCNFG. In Windows 2000 and later there is a components control panel that can be launched with a mouse. These tools allows you to select the application component to manage, set the location of the component (local or on a remote host), and choose permissions for launch, access, and configuration. In addition, you can specify the identity of the user that the component will run as. This allows the component to impersonate a user with more authority than the launching user. Security is applied using the local or domain groups and accounts available to the server.

DCOM Architecture

The following illustration from an article titled “DCOM Technical Overview” (Microsoft Corporation, 1996) gives a graphical view of DCOM architecture:



Several features of the DCOM architecture make it programmer friendly. The architecture is designed to allow easy reuse of existing components to provide a measure of location independence and to allow for some degree of language neutrality. Connection management, network efficiencies, and a built-in security structure complete the package.

Component reuse is leveraged with a COM/DCOM model; basically a developer can write a component one time and use it in multiple applications. This approach has the added benefit of reducing debugging time as existing components have already been debugged allowing the developer to focus on debugging the new code.

Location independence allows the developer and system administrator to more easily specify where components exist, where they run, and helps them plan their network to allow for the greatest efficiencies.

Language neutrality refers to the fact that DCOM components can be created from many different languages and tools among them are Java, Visual C++, Visual Basic, and several others. Programs written in one language can easily talk to and use the features of components written in another language. Best of all, this is transparent to the developer and to the system administrator.

DCOM handles all facets of connection management, keeping track of connections from one client to another and handles multiple connections from multiple clients to a single component. DCOM also keeps connections alive with an efficient periodic ping, and notifies applications of connections which terminate unexpectedly.

DCOM uses a technique called batching (or boxcarring) to reduce the amount of network round trips needed to compete a given series of application

communications. It also allows for several methods to custom tailor these communications for added efficiency.

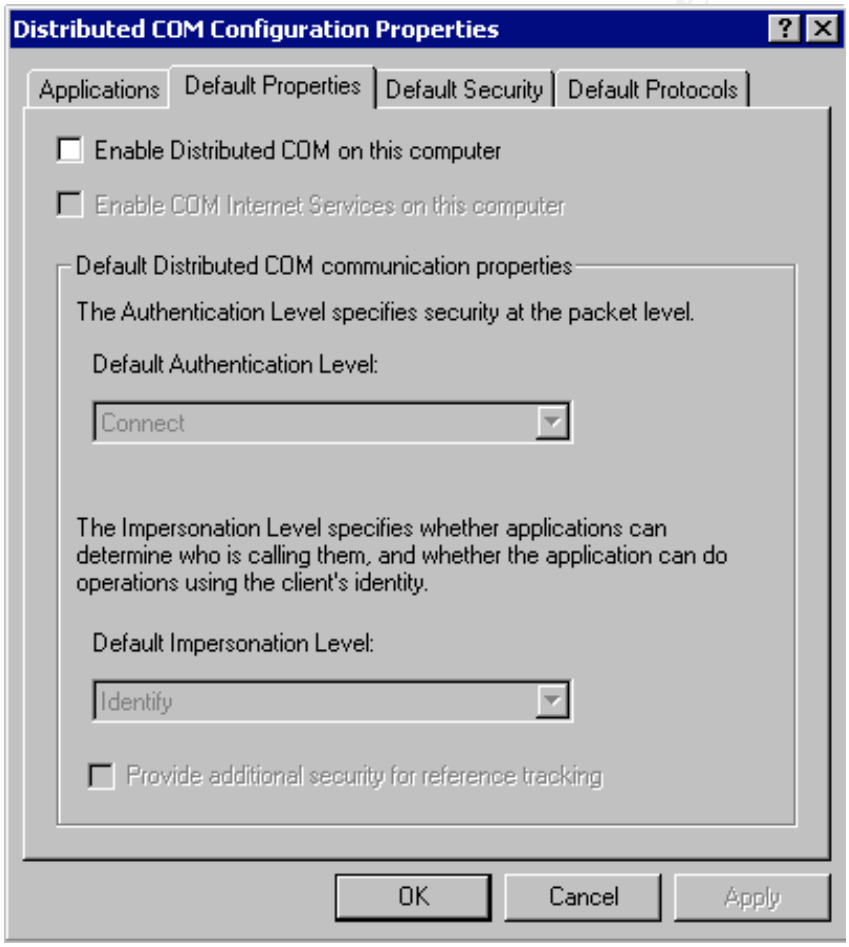
Administering DCOM Components

In the DCOM security model offers a great deal of flexibility for configuring who can launch a component and who the component should run as (Impersonate).

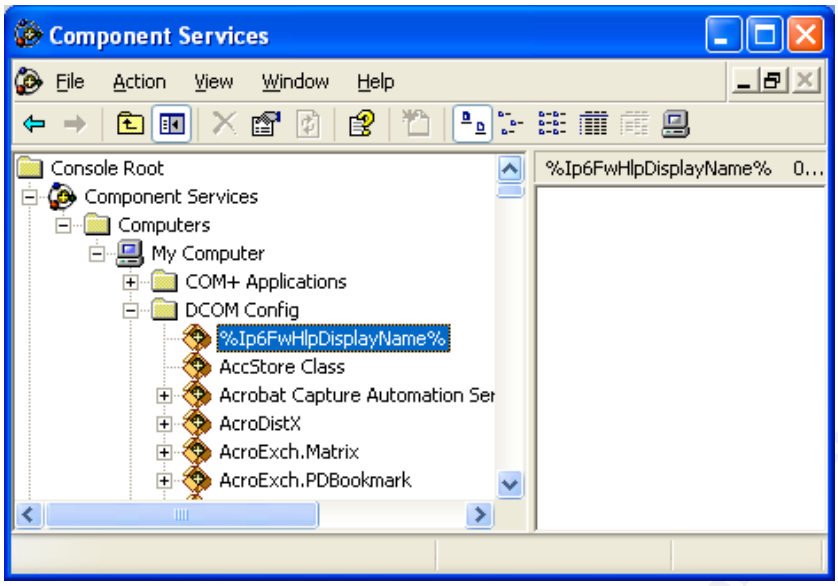
Examples:

- The component can be launched anonymously but run under the context of a domain account.
- A component can be configured to require credentials for launch and can run under the context of those same credentials.
- A component can be launched requiring the credentials of a member of a specified group, then impersonate an Administrator when running.

Screen shots of the DCOMCNFG tool:



The same tool in Windows 2000 and later uses an MMC snap in:



Walk Thru:

General

Launch the DCOM admin by going to the command line in and typing DCOMCNFG. The first Tab across the top is General. From the General tab you configure the Authentication Level. Authentication level defines the place at which authentication takes place:

1. Default – Object inherits machine default Authentication level.
2. None – No authentication is required or used.
3. Connect- Authentication is completed once at the start of the first connection
4. Call – Authentication is completed on every connection from start to finish of the session.
5. Packet – Authentication is completed for every packet.
6. Packet integrity – Same as packet level, with packet check-sums added
7. Packet privacy – encrypts all data

Select the appropriate level of authentication from the above choices. Items 1-4 are subject to hijack type attacks, item five is susceptible man in the middle type attacks, level six can still be intercepted and the data read, and level seven may be susceptible to brute force attacks depending upon the encryption used.

Location

The location tab is just like it sounds; it defines where the DCOM object actually runs.

- Run application on the computer were data is located
- Run application on this computer

- Run application on the following computer

From here, an administrator performing an application installation can define that a component will actually be executed on another computer.

Security

The security tab defines launch permissions, access permissions, and configuration permissions for each DCOM component on the system.

The administrator is given a choice to use the default machine settings or to customize and select Windows users or groups for each of the preceding settings.

Endpoints

Endpoints are used to configure a COM application to use a specified TCP port number for DCOM communications. The default is dynamically assigned ports.

Identity

The Identity tab allows an administrator to define which user context of the application will run in. Choose from:

- The interactive user – This will have the component run under the context of who ever happens to be logged into the server at the moment (could be nobody)
- The launch in user – This tells the component to run as the user who called the component.
- This user – This choice allows an administrator to select a specific account that the component will run as.

There is a final choice of the 'system account' which applies to services only.

DCOM Security

DCOM allows application developers to leverage Windows NT's built-in security framework. A detailed discussion as to the pros and cons of Windows NT's security are out of the scope of this document. However, I think most would agree; using a centralized security model is better by far than a requiring each software developer to devise and implement his own security scheme.

The first method for implementing security is security by configuration. The tools for managing this were briefly outlined in the previous section "Administering DCOM Components". Administrators installing DCOM applications can configure who has access to the components and who the components run as (impersonate). The operating system handles the verification of the user credentials and reports back to the application whether the calling

client is authorized or not. Unauthorized clients cannot even instantiate a component.

DCOM also offers programmatic control over security. Programmers can implement method by which the calling user's credentials are passed to the component doing the work. These credentials can then be used to open a file or Registry key that has been secured with regular NT permissions. In this manner a client can have access to a component, may be able to execute some of its methods, but may be denied access to others.

The NT security framework also offers methods for passing credentials outside of one's local computer domain. Built-in support for Kerberos authentication protocol, distributed password authentication, and secured channel services allow for the possibility of a widely distributed application that spans companies and NT domains.

Load-balancing

DCOM offers support for both static and dynamic load-balancing. With static load-balancing, clients can be pre-assigned to use certain servers. Windows 2000 implements a distributed class store which adds centralized management to these features.

Another approach to static load-balancing is a dedicated referral component. All clients looking for a particular resource connect first to the referral component and are then directed to the server which best suits its needs.

Next, to take load balancing a step further, applications can be designed to dynamically load balance which allows for scaling and configuration changes on-the-fly based upon load. With this approach, components are used to dynamically direct traffic for other components or components use Microsoft Transaction Server to assist with load balancing.

DCOM in Action

To create a better understanding of what DCOM is and what it can do, I will describe how I have used DCOM recently to solve two real world problems.

Problem 1: Web Hosting

I have a small Web hosting business. I needed to provide a way that users on the World Wide Web could sign up, provide contact and payment information, and have a web site created automatically. The web site at this point would just be an empty shell but would allow the user to log in using their newly created credentials, upload their content and display it on the World Wide Web without any operator or administrator intervention.

This problem presented several challenges:

1. Many of the functions described needed to be performed under the context of an administrator or privileged account.
2. The web site where users could sign up for hosting services was not necessarily on the same server as where to hosting service was to be provided.
3. The creation of the site for the end-user needed to be performed in the background transparently.
4. These actions needed to be completed in an environment where security must be managed tightly as the servers involved were accessible on the World Wide Web.

The answer to these with challenges was DCOM. Using DCOM I could accomplish all of the goals while maintaining the site's security.

The solution has several different parts. For the first part, I needed a way to create an NT account for the user to use to log in with, I needed to create end-user directories for the end-user's Web code with permissions appropriate for the end-user's account to be able to read and write to them, and I needed to create the actual web site inside of IIS which would host the web code.

I was able to write a small component which used Microsoft's Active Directory Services Interface (ADSI) to do most of this work. This component did everything that I needed for it to do, but, it needed to run under the context of an administrator.

I then created two accounts, one an unprivileged account, and one that was a member of the administrator's group. At the web site where users would come to signup I changed the configuration so that it ran under the context of the newly created unprivileged account. Then, using DCOMCNFG, I modified the launch and impersonation properties of my new component. I allowed only the new unprivileged account permissions to launch the component and, once launched, I configured the component to run under the context of the newly created privileged account.

I used DCOMCNFG on each computer involved to allow the web site to be created on a different server that was used to manage the sign up process.

This layered approach allowed me to accomplish my goals. My signup web site was the only website to use the newly created unprivileged account, and that account was the only account that had permissions to launch my component that did the actual work.

Even though my component runs under the context of an Administrator, it can only perform those functions that are built into it. It can't do anything but what it was designed to do. Even if someone, somehow, were able to hijack this

component, all it can do is create websites – and it notifies me each time it does so.

Problem 2: Credit Card Processing

Problem number 2 was a bit different. I was working for a company with a small local area network in their office. The "Processing Station" was a workstation on the Local Area Network (LAN) with a proprietary credit card processing software package. This workstation had an interface where a user could enter credit card and billing information for a transaction. The software would then take this information create a file and transmit it via a modem to a credit card processor which would approve or deny the transaction. The result would return a file which the user interface would read and provide the Approval or Denial to the operator.

This scenario worked perfectly until it became necessary to perform this credit card processing function on multiple computers on the LAN. Once this requirement was introduced we had to take action to provide this functionality. We needed a way to maintain the central processing station, but add the user interface to multiple, local computers.

DCOM provided a solution. Using DCOM we were able to create a component that would accept input from a user interface. Based on the input received the component would create the file needed by the processing station to transmit to the credit card processor and get an approval. The response would be processed and the result sent back to the originating station.

Each remote station had a user interface that was connected to a DCOM component that would do the work. The DCOM component connected to the processing station and did the work that was needed and returned the result to the remote station.

In this example, I created the DCOM component and built in only the functionality that I wanted the remote users to be able to use. Then we allowed each workstation to launch the component with the credentials of the logged in user. This allowed us to keep track of which users were performing which functions. This approach allows us to track suspect transactions back to the user that performed them. The processing station is protected as it does not require remote users to have direct access to the files created or processed on it.

DCOM and the Firewall

Most Internet applications have a fixed TCP and/or UDP port. A firewall administrator identifies the application that supposed to run on a particular machine, opens up a port to all users or to specific hosts then monitors for suspicious activity.

DCOM works a little differently. DCOM dynamically assigned ports at run time - one TCP port and one UDP port for each process, for each component, running on a server. Clients connect to the server first on port 135 and are directed to the proper port for the component that they are seeking.

Opening up port 135 on a server to the Internet is a scary thought. I think most firewall administrators would agree that opening ports 135-139 in anything but a very limited fashion should be avoided whenever possible. There are just too many things going on in any Microsoft based server on any of these ports to allow for proper security. Further, with this dynamic port assignment, a firewall administrator would have to open up all sorts of ports in both inbound and outbound directions. Negotiating on port 135 would only be the beginning. Next, of course, both host and client would need to be allowed to communicate on whatever port was dynamically assigned.

Dynamic port assignment might be an acceptable solution for local-area network but for Internet applications fixed port assignments are necessary. Fortunately DCOM has a method to allow for fixed port assignments. First, it is possible to configure the Registry of the server to allow TCP only connections. Next it's possible to configure the RPC service to use a pre specified port range. This is a one-sided fix performed only on the server clients would still connect up 135 and the server and direct them to its fixed port for the requested transactions. It improves security as it allows the firewall administrator to limit access to port 135 and the predefined range of ports. If server call backs are implemented, the server must be permitted to connect outbound on any port.

The fixed port assignment method CAN be used to provide for a secure computing environment. However, system administrators must be very careful to configure each and every DCOM component on the Internet accessible server, removing those which are not absolutely necessary, and securing those that remain. Keep in mind that the gateway to DCOM is port 135. Since port 135 must remain open to allow any DCOM functionality, and if ANY DCOM functionality is required then all DCOM components on a public server must be carefully scrutinized.

DCOM Concerns

I have a few concerns with the current implementation of DCOM. One concern is in the way the DCOM handles call-backs. A call-back occurs when a program tells a component to go and perform a task and report back if there is any trouble or when the task has been completed. This call-back method is called an event. A component is configured to “raise events”. These events can be received by the calling client. I have had much difficulty and have read through many reports regarding the use of events in an environment where the DCOM components have been secured. In reviewing the DCOM security architecture, it seems the functionality to support call-backs is present, but in

practice many developers and administrators have had a great deal of trouble in this area.

In the scenario presented, where an event notification back to the calling host is required it is very difficult to implement security on the DCOM components. To work around this apparent limitation, many resort to a much lower level of security or no security at all on their DCOM components.

Another concern is the apparent lack of knowledge and understanding by many system administrators as to what DCOM is and how to secure it. Even on a local area network there can be dozens of DCOM components on any given system and it is difficult to understand what each of them does and what risks that they pose. Many times a system administrator receives a set of instructions from a software company telling them how to configure the DCOM security for their component; usually a couple of screen shots and general recommendation for access permissions. In some cases the instructions tell you how to remove just about all access restrictions from a company's component to allow it to function properly.

DCOM comes enabled by default on all Microsoft Server and Desktop operating systems since Windows 98. I have not found many tools for auditing DCOM components and their associated security. In this paper I have gone over some of the methods that are available for making DCOM based applications secure, but it is up to system and security administrators to do these tasks and to understand their implications. It is easy to lose sight of DCOM components and can be very difficult to audit and maintain them.

References

Microsoft Corporation. "DCOM Technical Overview". November 1996
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp (8/1/2003)

Michael Nelson. "Using Distributed COM with Firewalls". 3/19/1999
URL: http://www.microsoft.com/com/wpaper/dcomfw.asp (8/3/2003)

Markus Horst and Mary Kirtland. "DCOM Architecture". 7/23/1997
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomarch.asp (8/10/2003)

Mary Kirtland. "Object-Oriented Software Development Made Simple with COM+ Runtime Services" November, 1997
URL: http://www.microsoft.com/msj/1197/complus.aspx (6/5/2003)

Mary Kirtland. "The COM+ Programming Model Makes it Easy to Write Components in Any Language" December, 1997
http://www.microsoft.com/msj/1297/complus2/complus2.aspx (6/5/2003)

Peter Wright. Visual Basic 6 OBJECTS. Birmingham. Wrox Pres, Ltd, 1998. 131 - 155.

Christopher Blexrud. Professional Windows DNA. Wrox Pres, Ltd, 2001. 85-100

Simon Robinson. Professional ADSI Programming. Wrox Pres, Ltd, 1999.