Converting an Intranet Application from NT 4.0 to W2K

Kelly Mutters
Version 1.4b
Option 1
August 1, 2003

Abstract

The purpose of this paper is to document the process of securing a COM+ server package on Windows 2000, using the least privileged model. The paper will include the step-by-step process, standards, and procedures as well as lessons learned during migration regarding COM+, DCOM and the differences between MTS on NT 4.0 and COM+ on Windows 2000.

The documentation for migrating an application from NT 4.0 to Windows 2000 will include base security settings for the MTS packages in NT 4.0, the proposed base security settings for the COM+ components in Windows 2000, working with the application owner to gather security requirements so that the least privileged model could be enforced.

Definitions

COM+ is an extension of Component Object Model (COM), Microsoft's strategic building block approach for developing application programs. COM+ is both an object-oriented programming architecture and a set of operating system services. It adds to COM a new set of system services for application components while they are running, such as notifying them of significant events or ensuring they are authorized to run. COM+ is intended to provide a model that makes it relatively easy to create business applications that work well with the Microsoft Transaction Server (MTS) in a Windows NT or subsequent system. It is viewed as Microsoft's answer to the Sun Microsystems-IBM-Oracle approach known as Enterprise JavaBeans (EJB).[1]

"Least privilege Security Model" is the definition of rules, roles and/or policies that allow for an organization to implement the most secure environment for their specific needs. The least privilege concept means a user is given only the access needed to fulfill their job requirements. [2]

Everyone Access grants any user the rights to resources, either locally or via the network, without authenticated credentials.

Authenticated User Access grants any user who directly or indirectly utilizes the Windows 2000 Server Integrated Sign-On Service or receives credentials from the Windows 2000 Directory Services.[3]

ACLs - a table that tells a computer operating system which access rights each user has to a particular system object, such as a file directory or individual file.[4]

Role Based Security is the process of setting security access at the Package and Component level using Active Directory objects.

DCOM - is a protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner.[5]
Subinacl - is a powerful and useful command-line tool (available in the Microsoft Windows 2000 Server Resource Kit and the Microsoft Windows NT Server 4.0 Resource Kit) that lets you directly edit almost any security information—permissions, ownership, or auditing—on objects of all kinds. Not only can you use Subinacl to modify this information on files, directories, file shares, and printer shares, you can also use the tool to control permissions on registry keys, system services, and the Microsoft IIS metabase.[6]

MTS on NT 4.0

The NT 4.0 environment contains two separate servers for Intranet Applications. The two servers house library and server packages. The library packages should be used for trusted transactional components, in which no additional security is required and server packages should be used for untrusted components, transactional or not.[7]

COM+ on Windows 2000

The Windows 2000 environment brought changes for Intranet applications. MTS was changed to COM+ and the load of the two servers was combined into one.

Windows 2000 offers more granular security using the least privileged model. With the limited granularity in NT 4.0, most of the time users that needed elevated privileges to perform their job, had to be given administrative access. Since the least privilege model is based on requirements and administrative access is not needed for elevated privileges, moving to the least privilege model is often very painful for an organization. It is the job of the Security Analyst to take advantage of the granularity offered in Windows 2000. The Security Analyst also needs to make sure that the application owner does not feel that they are no longer trusted with administrative privileges or that they have done something wrong.

Implementation of the least privileged security model does not come without cost. More hours and more expertise are needed to make the applications function correctly with minimal required accesses.

Process for creating a secure COM+ application in Windows 2000

The first step in the process of securing a COM+ application in Windows 2000 using the least privileged model is a meeting with the application owner to define the security requirements. In order to gather these requirements, the application owner needs to provide the following:

- The COM+ package name
- Any components associated with the package

- The directory structure the application uses
- The users who need access to the directory structure
- The types of access the users need to the directory structure
- The registry key the application uses(if any)
- The users who need access to the registry
- The type of access the users need to the registry

The second step in the process is the creation of a security script. The purpose of the script is as follows:

- Enforce access checking for the application. In order to minimize the security exposures for COM+, it is best to require security to be enabled on all COM+ packages. To enforce this standard, the security script should enable authorization on the COM+ packages to enforce security at the package level.
- Set the unique identity to the package using an Application Account.

The application account associated with a specific application can be local to the server or a domain account, depending on the requirements of the application. The following rules should be incorporated within the security script:

- Grant the appropriate access roles at the package level
- Grant the appropriate access roles at the component level
- Set file ACLs
- Set registry ACLs

  Note: To minimize security risks for the entire organization, if the application account has high security exposures, the application account should be local to the server. Should the account be comprised, only that specific server is exposed instead of the entire network. If the application account has a low security exposure, then a domain account should be used. Application accounts should not be used as generic IDs to access network resources.

  Suggestion: All application accounts must identify the exact name of the application(s) that is running under the account, all impacted servers that use this account, a brief description of what the application(s) is doing, and a password management plan. This plan would provide the step-by-step process in changing a password if it becomes necessary months or years later due to a security exposure.

The roles will contain application and user accounts that need access to the application. The application and user accounts also need access to the associated directory containing the application dlls in order for the intranet transaction to complete successfully.

Common Issue: Most application owners are unsure who actually uses their application.

Solution: The most secure intranet application is one that is locked down to specific users. However, when it is not possible to define each user individually and the data is not classified as sensitive or confidential, the application and associated directory containing the dlls read access is given to authenticated users.

Note: Granting access to the authenticated users group is more secure than granting access to the everyone group.

In order to execute a package's dll, the user will need Read and Execute access rights to the dll. You can manage ACLs at the directory or file level. Usually the standard for large directory structures is to ACL at the directory level unless a requirement for a specific file indicates different access then the parent directory. The cost of maintaining ACLs at a file level are significant and should only be used when requirements dictate. If the package does not contain data that is classified as confidential or sensitive, Authenticated Users should be granted read and execute access. If the requirements dictate that the package is confidential or sensitive, the data should be protected by a group containing specific user ids having Read and Execute permissions.

Role Based Security

Role Based Security should be used to enforce the security settings on the COM+ server packages. In order to comply with the Windows 2000 least privileged security model, server packages should have at least one role associated with them even if the role only contains "authenticated users".

The following items should be considered when using Role Based Security:

- Role names are case sensitive. When scripting role names, this can cause failure if Active Directory contains a different casing for the role name than the script.
  Suggestion: Create a standard stating the naming convention for all roles.
- In order for the IsSecurityEnabled to return True, both the component and the package must have security activated.
  Note: This is a change from MTS on NT 4.0 in which IsSecurityEnabled always returned True.
- When a component is installed under a package, by default, security is enabled with no roles assigned. In order for the component to function, a role needs to be assigned to the component or security needs to be turned off at the component level.
  Suggestion: For component level security, assign a role to the component only if requirements determine that the component needs to be secured, otherwise, the "Component level access security"

should be unchecked.

Suggested Standards to enforce for consist and security COM+ Packages

The following standards will help expedite troubleshooting during the testing and implementation of COM+ packages:

1. Assign every package a unique package id.
    - Prefix the package id name with COM_ for easy identification of all COM+ accounts in active directory.
    - Use a naming convention that easily identifies the package.
2. Enable Security on all COM+ Packages.
3. If requirements dictate, enable roles for components.
4. Use Domain Local Groups for setting the roles.

Testing Procedures for COM+ Packages

Thorough testing of the security for COM+ packages in Windows 2000 is essential for the success of implementation. Three levels of test gain the most benefit during the implementation of COM+ applications.

The first level of test

The first level of test should verify functionality of the application in the Windows 2000 environment and confirm that the security requirements for the application have been met. To test functionality of the application, it should first be loaded in the environment and then a user who has administrative access should launch the application to see if it functions properly. If the test is successful, then functionality of the application has been verified.

Next, the security settings obtained from the requirements meeting should be applied to the application. A user who has the security access defined through the requirements should launch the application to ensure no security errors are displayed. If the application still functions properly after the security has been applied, then the application should be marked successful and sent to the next level of test.

Note: Since the application functioned correctly with administrative access, the security analyst needs to work closely with the application owner to determine what security requirements need to be changed in order for the application to function properly.

The second level of test

The second level of test should verify functionality with multiple applications and other systems. This level of test should load the application with security and then have a user with the correct access test the application. This test is performed in the same forest as the first level. If this test is successful, the application should be marked successful and sent to the final level of test.

The third level of test

The third level of test should give the final verification that the application is functioning properly before being implemented in production. Once again all software components should be loaded with security applied and then the user with the correct access should test the application. This test is performed in a different forest then the first two levels of test to mimic production. If any part of the testing fails in this level of test, then it should go back to the first level of test where problem resolution needs to occur and then the process starts all over again.

Once the testing is successful at this level, then the application is ready to be implemented in production following the same steps used in the second and third levels of testing.

Troubleshooting Tips

If testing fails in any of the three environments, then the following tips for troubleshooting can be used for resolution. The first item to troubleshoot should be the Default DCOM Security.

DCOM was configured differently between NT 4.0 and Windows 2000. The following steps can be used to verify the configuration of DCOM in Windows 2000:

1. Logon to the server using an account that has administrative access
2. Click on the start menu , select run and type dcomcnfg
   Note: this will open the Distributed COM Configuration Properties window.
3. Select the Default Security tab
4. Under the Default Access Permissions, select the Edit Default button.
   Note: this will open a Registry Value Permissions window which displays the users\groups and their permissions.
5. Click the Add button
   Note: this will open the Add Users and Groups window
6. The List Names field has a pull down menu, select the local server name
   Note: it should appear at the top of the menu list
7. In the Names window, select the user INTERACTIVE, click Add, click OK. This will close the Add Users and Groups window.
8. Verify that the user INTERACTIVE is now listed in the Registry Value Permissions window with Allow Access permissions. Click OK to close the Registry Value Permissions window.

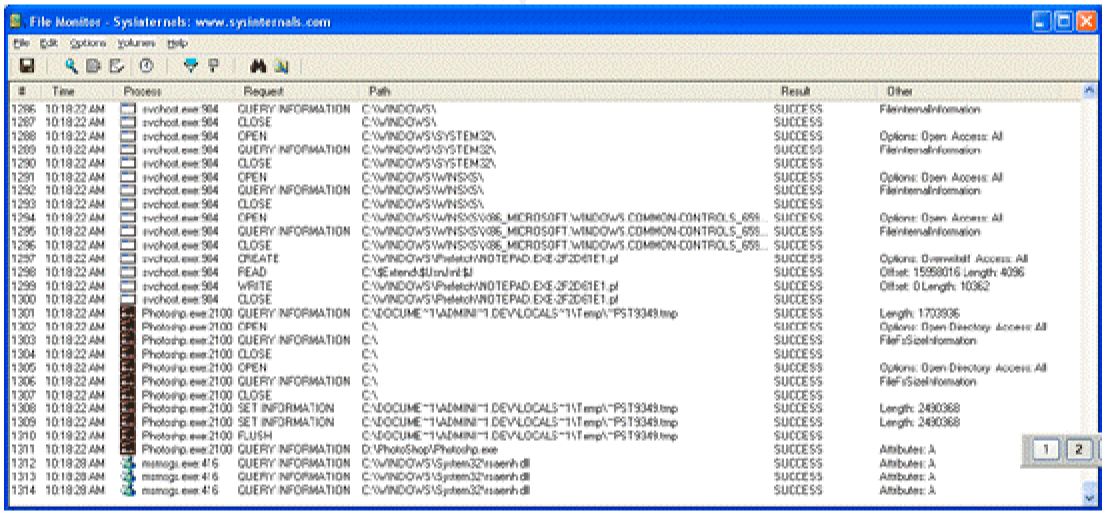
9. Click OK to close the Distributed COM Configuration Properties window.
10. Logoff the server.

If the Default DCOM security is correct and the application stills has security issues associated with it, then the Filemon utility can be used to try and pinpoint the problem.

Filemon monitors and displays file system activity on a system in real-time. Its advanced capabilities make it a powerful tool for exploring the way Windows works, seeing how applications use the files and DLLs, or tracking down problems in system or application file configurations. Filemon's time stamping feature will show you precisely when every open, read, write or delete, happens, and its status column tells you the outcome. Filemon is so easy to use that you'll be an expert within minutes. Filemon begins monitoring when you start it, and the output window can be saved to a file for off-line viewing. Filemon has full search capability, and if you find that you're getting information overload, simply set up one or more filters.[8]

The following steps start the Filemon process:

1. Start Filemon
2. Execute application
3. Stop Filemon
4. Review Output



```
#     Time         Process              Request             Path                                                            Result    Other
1286  10:18:22 AM  svchost.exe:984      QUERY INFORMATION   C:\WINDOWS\                                                     SUCCESS   FileInternalInformation
1287  10:18:22 AM  svchost.exe:984      CLOSE               C:\WINDOWS\                                                     SUCCESS
1288  10:18:22 AM  svchost.exe:984      OPEN                C:\WINDOWS\SYSTEM32\                                            SUCCESS   Options: Open Access: All
1289  10:18:22 AM  svchost.exe:984      QUERY INFORMATION   C:\WINDOWS\SYSTEM32\                                            SUCCESS   FileInternalInformation
1290  10:18:22 AM  svchost.exe:984      CLOSE               C:\WINDOWS\SYSTEM32\                                            SUCCESS
1291  10:18:22 AM  svchost.exe:984      OPEN                C:\WINDOWS\WINSXS\                                              SUCCESS   Options: Open Access: All
1292  10:18:22 AM  svchost.exe:984      QUERY INFORMATION   C:\WINDOWS\WINSXS\                                              SUCCESS   FileInternalInformation
1293  10:18:22 AM  svchost.exe:984      CLOSE               C:\WINDOWS\WINSXS\                                              SUCCESS
1294  10:18:22 AM  svchost.exe:984      OPEN                C:\WINDOWS\WINSXS\X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_659...  SUCCESS   Options: Open Access: All
1295  10:18:22 AM  svchost.exe:984      QUERY INFORMATION   C:\WINDOWS\WINSXS\X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_659...  SUCCESS   FileInternalInformation
1296  10:18:22 AM  svchost.exe:984      CLOSE               C:\WINDOWS\WINSXS\X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_659...  SUCCESS
1297  10:18:22 AM  svchost.exe:984      CREATE              C:\WINDOWS\Prefetch\NOTEPAD.EXE-2F2D61E1.pf                     SUCCESS   Options: OverwriteIf Access: All
1298  10:18:22 AM  svchost.exe:984      READ                C:\$Extend\$UsnJrnl:$J                                          SUCCESS   Offset: 15958016 Length: 4096
1299  10:18:22 AM  svchost.exe:984      WRITE               C:\WINDOWS\Prefetch\NOTEPAD.EXE-2F2D61E1.pf                     SUCCESS   Offset: 0 Length: 10362
1300  10:18:22 AM  svchost.exe:984      CLOSE               C:\WINDOWS\Prefetch\NOTEPAD.EXE-2F2D61E1.pf                     SUCCESS
1301  10:18:22 AM  Photoshp.exe:2100    QUERY INFORMATION   C:\DOCUME~1\ADMINI~1.DEV\LOCALS~1\Temp\~PST9349.tmp             SUCCESS   Length: 1703936
1302  10:18:22 AM  Photoshp.exe:2100    OPEN                C:\                                                             SUCCESS   Options: Open Directory Access: All
1303  10:18:22 AM  Photoshp.exe:2100    QUERY INFORMATION   C:\                                                             SUCCESS   FileFsSizeInformation
1304  10:18:22 AM  Photoshp.exe:2100    CLOSE               C:\                                                             SUCCESS
1305  10:18:22 AM  Photoshp.exe:2100    OPEN                C:\                                                             SUCCESS   Options: Open Directory Access: All
1306  10:18:22 AM  Photoshp.exe:2100    QUERY INFORMATION   C:\                                                             SUCCESS   FileFsSizeInformation
1307  10:18:22 AM  Photoshp.exe:2100    CLOSE               C:\                                                             SUCCESS
1308  10:18:22 AM  Photoshp.exe:2100    SET INFORMATION     C:\DOCUME~1\ADMINI~1.DEV\LOCALS~1\Temp\~PST9349.tmp             SUCCESS   Length: 2490368
1309  10:18:22 AM  Photoshp.exe:2100    SET INFORMATION     C:\DOCUME~1\ADMINI~1.DEV\LOCALS~1\Temp\~PST9349.tmp             SUCCESS   Length: 2490368
1310  10:18:22 AM  Photoshp.exe:2100    FLUSH               C:\DOCUME~1\ADMINI~1.DEV\LOCALS~1\Temp\~PST9349.tmp             SUCCESS
1311  10:18:22 AM  Photoshp.exe:2100    QUERY INFORMATION   D:\PhotoShop\Photoshp.exe                                       SUCCESS   Attributes: A
1312  10:18:28 AM  msmsgs.exe:416       QUERY INFORMATION   C:\WINDOWS\System32\rsaenh.dll                                  SUCCESS   Attributes: A
1313  10:18:28 AM  msmsgs.exe:416       QUERY INFORMATION   C:\WINDOWS\System32\rsaenh.dll                                  SUCCESS   Attributes: A
1314  10:18:28 AM  msmsgs.exe:416       QUERY INFORMATION   C:\WINDOWS\System32\rsaenh.dll                                  SUCCESS   Attributes: A
```

If the output reveals that the user is trying to access a file or DLL that does not have the appropriate access, the security script needs to be updated to include the needed access.

Note: Since the Least Privileged Model should be used while migrating to Windows 2000, standards need to be developed and followed when granting access to files and dlls that belong to another application. When access is being granted to another application's files or dlls, a written confirmation from the owner of the application in which the additional access is being granted should be obtained.

If the filemon utility does not reveal that additional security is needed to a file or dll, then checking access on the registry is the next step. A utility that can assist on identifying additional registry access settings is Regmon.

Regmon some light on the mystery of the OS by showing you which applications are accessing your Registry and which keys they are accessing. It shows you all of the Registry data your programs are reading and writing. And Regmon does all this in real-time: Unlike other tools, which show you after the fact what values and keys have changed, Regmon shows you the changes while they are happening. One thing is certain--after using this tool, you'll never think of your OS the same way.[9]

The following steps start the Regmon process:

1. Start Regmon
2. Execute application
3. Stop Regmon
4. Review Output

If the output indicates the application is trying to access a registry key it does not have access to, the security script will need to be updated to allow proper access to the identified registry key.

Note: Once again, since the Least Privileged Security Model should be used when migrating to Windows 2000, standards need to be developed and followed when granting access to registry keys. Each registry key should have an owner and that owner should supply written approval of another application needing additional accesses to their registry key. This written approval serves two purposes:

1. To make sure the application owner understands that another application will be given access to the registry key associated with his application.

2. Should something happen to the registry key in which additional access was given, documentation confirming which applications have access is on file.

Subinacl

If an ACL misordering error occurs when viewing the registry or file settings, the subinacl version should be identified. Microsoft subinacl versions prior to 2.6.0.1399 contains a bug that produces the misordering of ACLs.[10]

Summary

The purpose of this paper is to describe the most appropriate process to use when migrating an intranet application from NT 4.0 to Windows 2000. The practical is also intended to assist in troubleshooting security issues once the migration is complete. The troubleshooting tips listed are for assistance in identifying the root cause in the most logical and efficient manner.

References

1. Search WebServices.Com. 10 Jan 2001. http://searchwebservices.techtarget.com/gDefinition/0,294236,sid26_gci211825,00.html (1 Aug 2003)

2. ITsecurity.com. 8 March, 2003. http://www.itsecurity.com/papers/waveset1.htm (1 Aug 2003)

3. W2Kauthenicateduser.doc http://accsubs.unsystem.org/iscc-intranet/work/documents/doc/W2Kauthenticateduser.doc (1Aug 2003)

4. searchSecurity.com Definitions - powered by whatis.com. 23 Apr 2003. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci213757,00.html (1 Aug 2003)

5. Distributed Component Object Model http://www.microsoft.com/com/tech/dcom.asp (1 Aug 2003)

6. Windows and .Net Magazine. October 2002. http://www.win2000mag.net/Articles/Index.cfm?ArticleID=26362 (1 Aug 2003)

7. Microsoft Product Support Services http://support.microsoft.com/support/activeserver/27 (1 Aug 2003)

8. SysInternals http://www.sysinternals.com/ntw2k/source/filemon.shtml (1 Aug 2003)

9. PC World http://www.pcworld.com/downloads/file_description/0,fid,8088,00.asp (1 Aug 2003)

10. Microsoft Product Support Services. 4 Jun 2003. http://support.microsoft.com/default.aspx?scid=kb;en-us;296865 (1 Aug 2003)