Security Considerations for the Integrated File System (IFS)

By Carol Woodbury

When I talk to administrators about the top issues in IT security, one topic that seems to make them squirm is the Integrated File System (IFS). By now most people have heard of the IFS but few understand the security considerations one must make to ensure this part of their system is secure.

First, let me clarify what I mean by the term "IFS." Integrated File System or IFS is really the title given to a set of file systems that are available on OS/400. (See **Figure 1**.) Those file systems include the QSYS.LIB file system (which is the "traditional" OS/400 file system that we know and love), the NFS file system, the QNTC file system that supports the Integrated xSeries Server for iSeries and other Windows 2000 servers on the network. Other file systems may be available, depending on the features and products installed on the system. Each of these file systems use a different security scheme. However, the file systems that I am referring to when I use the term "IFS" are those file systems that follow a UNIX-based security scheme. These include root ('/'), QOpenSys and user-defined file systems.

Why should you care about the security implications of the IFS? Because it enables many of the most powerful and most used features on OS/400 and i5/OS. iSeries Access for the Web, Java, and WebSphere applications are all implemented in the IFS, not to mention the vast number of configuration files, including most of the configuration files for the TCP/IP servers that reside in the IFS as well as the mail that is processed through or stored in the IFS by the POP and SMTP servers and the list goes on.

Let's look at the areas that concern me the most:

Unfamiliar and Therefore a Challenge to Administer

Because the security schemes of the various file systems are different than they are used to, most OS/400 and i5/OS Security Administrators find it a challenge to administer them. Administrators need to learn the UNIX authority scheme to administer the IFS. UNIX uses the following authorities:

- *R (Read) – required to read the contents of an object, including listing the contents of a directory
- *W (Write) – required to add an object to a directory or update an object
- *X (Execute) – required to traverse a directory. In other words, to display a stream file in the following directory path, one would need to add *X to all the directories and subdirectories in the path: /Directory/Subdirectory_1/ Subdirectory_2/ Subdirectory_3/ File_name.stmf

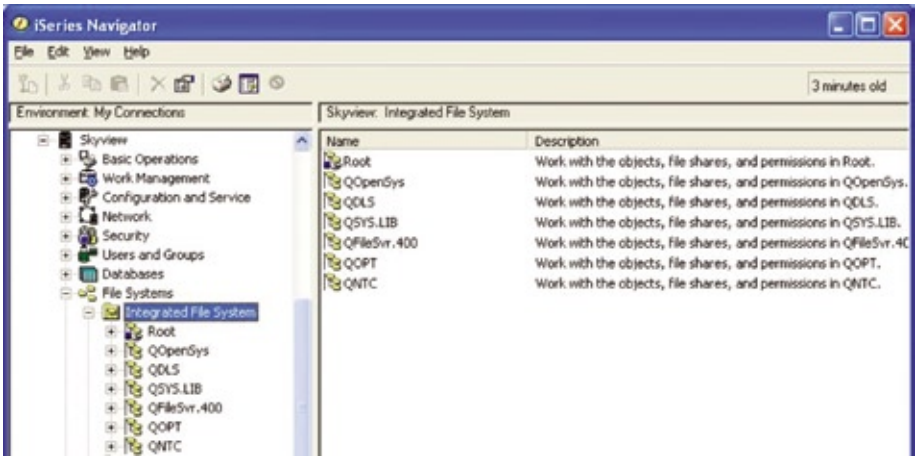

Figure 1. File systems in the Integrated File System (IFS)

Combine *RX and you get the equivalent of OS/400's *USE authority. Combine *RWX and you get the equivalent of *CHANGE. These are the data authorities. Unfortunately, when dealing with the IFS, you have two sets of authorities to manage. Despite the fact the you are dealing with these objects as if they were UNIX objects, you can't ignore the fact that underneath the covers, they're still implemented as OS/400 objects, so you must also deal with the "object authorities" – object management, object existence, object alter and object reference. In practice, the data authorities may get granular but the object authorities are usually *ALL or *NONE. You can manage these authorities through "green screen" commands (Change Authority (CHGAUT) and Work with Authority (WRKAUT)), by taking option 9 off of the Work with Object Links (WRKLNK) command once you've navigated to the object or through iSeries Navigator by right-clicking on the object and choosing Permissions. If you're using either CHGAUT or WRKAUT, you'll have to be prepared to type in the entire pathname for the object name. See **Figure 2**.

Note: When you administer authorities on IFS objects, make sure you administer both the data authorities (DTAAUT) and the object authorities (OBJAUT). I have seen instances where the administrator adjusted the data authorities but didn't realize he also needed to manage the object authorities.

Default Values and How They Should Be Set

Unfortunately, as root ships, it leaves a rather wide hole in the security configuration of the system. Root ships with the equivalent of public authority *ALL. That is, data authorities of *RWX

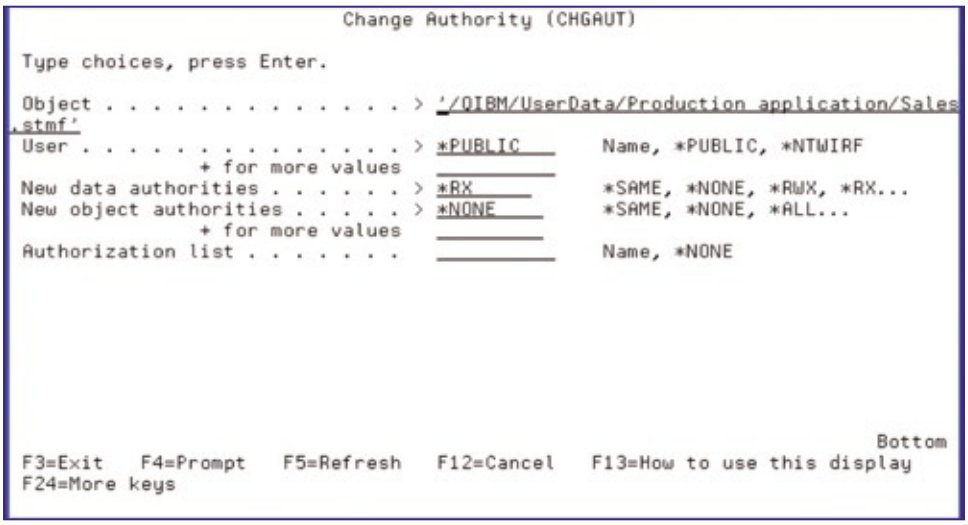

Figure 2 – Changing the authority on an IFS object using CHGAUT

and object authorities of *ALL. This setting allows anyone to create a directory directly under root and store whatever they want in this directory – and they do! I have seen directories with PC back-ups, pornography, other images and graphics downloaded from the Internet and illegal copies of movies stored in these directories. As an administrator, you will want to take control of who can create directories, just like you take control of who can create a library. Therefore, you will want to restrict access to the Create Directory (CRTDIR) and Make Directory (MKDIR) commands.

The wide-open access of root is compounded whenever a directory is created into root. When a directory is created it usually inherits the authority of its parent directory. (The exception is when you create IFS objects through APIs such as mkdir(), open(), or creat(). With those APIs, you can specify the data authorities for the owner, primary group, and public authorities.) The same is true when stream files, text files or other objects are created into a directory.

So the wide-open definition of '/' is continually propagated. It also means that anyone on the system can update or delete inappropriately secured file system objects – hardly the control a security administrator needs to ensure a stable and available system and accurate production data.

Managing Authorities

Let's take a look at some of the tools that are available to make managing IFS authorities easier:

The Print Private Authority (PRTPVTAUT) and Print Public Authority (PRTPUBAUT) commands were enhanced to include directory and stream file objects. These commands are the easiest way to get the total picture of your IFS authority structure. Beware however, if you specify to include all sub-directories under '/', the resulting report can be quite large and overwhelming.

The QPWFSERVER authorization list has been shipping with OS/400 for quite some time, but few people utilize it. It ships with public authority set to *USE. If you change that value to *EXCLUDE or specifically exclude a user or group, then access to the QSYS.LIB file system will not be allowed through directory interfaces such as Windows Explorer.

In other words, even if the user has mapped a drive and the file share includes the QSYS.LIB file system, the user will be prevented from accessing the QSYS.LIB file system through Windows Explorer unless the user has at least *USE authority to the QPWFSERVER authorization list.

Some challenges remain, however. While it is quite easy to change the owner (using the Change Owner (CHGOWN) command) or change the public authority of all of the objects in a directory (using the Change Authority (CHGAUT) command), the only way to propagate either of these operations through to any subdirectories is to write a program or perform the operation manually for each sub-directory.

File Shares

File shares are what makes a file system or a directory within the file system available for viewing or manipulation via the network. File shares allow users to map a drive to the directory, making the directory appear as part of the PC directory structure when viewed from an interface such as Windows Explorer. AS/400 ships several file shares. The exact number depends on the features installed.

To create a new file share use iSeries Navigator. Go to My Connections → iSeries_name → File Systems → Integrated File System. Right click on the directory or file you want to share. Choose Sharing.

Shares can be defined as read-only or read/write. Obviously, the more secure setting is read-only because – as the name implies – users can only read the data which is being shared on the network and not update it. However, whether the share is defined as read-only or read/write, OS/400 security has the ultimate say over whether the action is taken. If a read-only share is defined for a directory and John maps a drive to the directory, if John is excluded from the directory, he will not be able to see the contents of the directory. Likewise, if the share has been defined as read/write, John would have to have *W (*write) authority to the directory to add additional objects to the directory.

While a convenient way to share data throughout a corporation, defining shares can create serious security exposures. For example, defining a read/write share for root provides access to the entire directory structure of OS/400 via the corporate network. This means that any user that can map a drive to OS/400 or has a symbolic link for root defined on their desktop can access any file on the system that has *PUBLIC authority set to at least *R (read authority) or *USE authority. On many systems, this means that most, if not all files on the system are accessible. In many cases public access allows anyone with an OS/400 user id and password to download, update, replace or, in some cases delete OS/400 objects.

To see the existing shares, click on File Shares under File Systems in iSeries Navigator. Right click on a share and

choose Properties to see whether the share is read-only or read/write. When viewing directories or files under Integrated File System, shares are indicated by a hand underneath the directory or file name. See **Figure 3**.

Summary and Recommendations

1) Review the *PUBLIC authority for application and user directories. Approach the securing of directories the same way libraries are secured. A great deal of security can be gained simply by securing the library and authorizing only those users needing access. This same approach also works well for directories. That is, exclude the general public from accessing the directory and authorize only those users with a business requirement.

2) Review (and Remove) File Shares. Most systems that I have seen have an over-abundance of file shares. In fact, most systems have a file share defined for root '/' which makes the entire IFS available to the network. Many of these shares have been added by programmers to check out some of the latest and greatest programming tools. Some have been added to allow sharing of data but in almost every case, the shares have been added without thought to what is being made available to the entire network and usually, without the knowledge of the security administrator. I highly recommend that you review the file shares that have been defined, understand the security implications and remove the ones that are not appropriate.

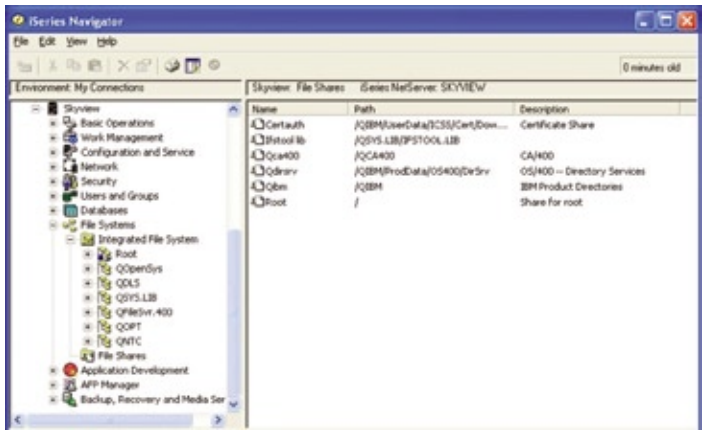

Figure 3: The hand under the Root directory in the left hand nav pane indicates that a share has been defined. The share is shown on the right.

Carol Woodbury *is co-founder of SkyView Partners, Inc, a firm specializing in policy management and assessment software as well as security consulting and services. Carol is the former Chief Security Architect for AS/400 for IBM in Rochester, MN, USA and has specialized in security architecture, design and consulting for over 15 years. Carol speaks around the world on a variety of security topics and is co-author of the book,* Experts' Guide to OS/400 and i5/OS Security.