SECURITY PATROL

by Vincent LeVeque

Conflicting Authority

QUESTION: If a user profile has one level of authority specified in an authorization list assigned to an object and a different level granted as a private authority to the same object, how is the difference reconciled? In other words, which authority rule takes priority?

ANSWER: It depends mainly on what version and release of OS/400 you are referring to but also on how you establish other security-relevant factors. OS/400 evaluates a user's authority to an object through a specific sequence of questions. At each step, a definitive answer, either "yes" or "no," will either permit or deny access, respectively, with no further checking. If a step does not provide a definitive answer, then OS/400 asks the next question. Here is the general order of questions:

1. Does the profile have ALLOBJ special authority? If so, the user gets access to the object, regardless of any other authorities which may have been defined.
2. Does the individual user have specific authority to the object, either through object authority or by being the owner of the object?
3. Is there an authorization list that grants or denies authority to the user?

If the user is a member of a group profile, then ask the following questions:

1. Does the user belong to a group profile that provides ALLOBJ? If so, then the user gains authority at this step.
2. Does the user's group profile have specific authority to the object, either though object authority or by owning the object?
3. Does an authorization list grant authority to the user's group?

If none of these steps provides a definitive answer, then PUBLIC authority for the object is reviewed.

The "joker" in this deck of cards concerns programs that adopt owner's authority. Adopted authority is examined last, after reviewing PUBLIC authority. However, if it results in the user gaining authority to the object, adopted authority takes precedence over any prior denial. This goes against the earlier rule of "first match gives result." Essentially, the entire process I've described is repeated for the program's owner (i.e., does the program owner have ALLOBJ?, etc.).

In your specific case, if the private authority results in a definitive grant or deny, then this will be the authority given to the user; whatever you might specify in the authorization list would not matter.

Stop! Don't Touch That Spool File!

QUESTION: I am working on a method of archiving specific spool files. We receive about 150 reports a day from our application that we need to archive, and the rest can be deleted. Once the report is created, we move it to specific output queues and then archive it into Magellan Software's SpyView, which is on a different AS/400. Users have been known to delete these spool files before I can get them captured, yet I cannot take away the Delete Spooled File (DLTSPLF) command to prevent this. I've tried securing the output queue object and even putting the queue into a secured library, but I've had no success because the owner of the spool file has access regardless of his rights to the queue. Is there a way to transfer ownership of the spool file or make a copy so that the original user can no longer access or delete it?

ANSWER: I assume you are unable to provide the necessary security by securing the DLTSPLF command. That's too bad: This would have been an elegant solution, otherwise. This approach would involve setting users' authority to all spool file management commands to *EXCLUDE. These commands would include DLTSLF as well as Work with Spooled Files (WRKSPLF), Work with Spooled File Attributes (WRKSPLFA), Change Spool File Attributes (CHGSPLFA), Hold Spooled File (HLDSPLF), and others listed in *OS/400 CL Reference V4R1* (SC41-5722-00). While your primary concern is preventing unauthorized spool file deletion, I assume you would want to control other changes to spool files, as well.

Your users may still require some carefully controlled access to spool file commands in order to view their own commands and, perhaps, print a copy for their own reference. I would suggest writing a short program to act as a front-end to the OS/400 spool file commands. This front-end program could determine a user's identity, his authority to use certain commands and options, and for which spool files he may use those commands and options.

Elegant solutions don't always work, and I must assume from your letter that you have tried this approach and it has failed in your environment. (I described the option anyway because it may work very well for other readers.) Your next option, given that the commands apparently cannot be secured, is to attempt to secure the spool file objects themselves. Here, you run into a very difficult problem, one caused by the nature of the AS/400's security model. The AS/400 supports something called *Discretionary Access Control* (DAC). DAC basically states that the owner or creator of an object has the final authority to decide who has access to an object. An owner always has the final say, short of revoking ownership and granting it to some other user. The usual alternative to DAC is *Mandatory Access Control* (MAC). In MAC, the organization's policies decide access rights for certain types of objects, and the specific user who created the object has no special rights at all. MAC is generally associated with a military-style security policy, classifying information as Secret, Top Secret, etc. and granting rights based on this classification. The AS/400 definitely does not support a MAC-based security policy.

To make your objective even more challenging, you can't get around DAC limitations by simply changing the object's owner for spool files. The Change Object Owner (CHGOBJOWN) command does not support spool file object types. There are some options on the OUTQ object for providing additional security, but these all specifically give the spool file owner full control of all the output that is created.

To draft a solution for you, I will assume your software follows good programming practices and does *not* generate reports interactively, given that doing so places a tremendous burden on the interactive subsystem and harms system performance for other users.. I assume that, instead, it submits a batch job that creates the report for the user.

The option I suggest is to create a new profile that will own all the spool files that your users generate. You should not assign this profile to any specific user, and it should have a password of *NONE to prevent anyone from using it to log in. It's sole purpose is to own spool files. For the sake of illustration, let's call this profile Bob. Make sure that Bob has all the authority that a user would normally possess as required for creating reports (this should include READ access to all application database objects). Now, let's assume there's a typical user with a profile named Alice. After creating Bob, grant Alice USE authority to the user profile Bob as follows:

```
GRTOBJAUT .....OBJ(QSYS/BOB) OBJTYPE(*USRPRF)
USER(ALICE) AUT(*USE)
```

The next step is to modify the Submit Job (SBMJOB) command that your software uses. Place the value Bob into the parameter USER. This will cause the submitted job to run under Bob's identity. Previously, you had to grant Alice (the actual user) USE authority to Bob's user profile object in order to permit this at QSECURITY level 40. Be careful here, as Alice might be able to submit unauthorized jobs under Bob's authority.

The end result of this is that Alice can verify the status of her print jobs using Work with Submitted Job (WRKSBMJOB) but cannot view any detailed job information or view or access in any way the spool files printed by the job. If she tries, she gets the message "No authority to job." In addition, your users are allowed to create spool files which they cannot delete, copy, or even view. To provide users access to their reports, you will need to do a bit of programming. Create a routine that runs the Copy Spooled File (CPYSPLF) command, creating a database file to which users will have read-only access. This program should run frequently enough to provide desired user access to reports. Users may view the reports through a simple program which reads the database file one line at a time and displays it in a scrollable window.

Freeware

QUESTION: I tried to go to the URL you provided for the Linux version of Strobe in the December 1999 "Security Patrol." All I got was an error 404 on this URL indicating that the link could no longer be found. Do you know where I can find the program?

ANSWER: I really hate it when this happens! You've just found one of the big costs of so-called free software on the Internet--the software may no longer exist by the time you try to go get it. Strobe was authored by the well-known Australian security specialist Julian Assange in 1995. It was originally hosted on his Web site at *suburbia.net*, but, apparently, this site is no longer available. In this case, the item is not hosted by the original author but is maintained in various standard software archives. If you use Red Hat Linux, the place to look is *rpmfind.net/linux/RPM/contrib/libc5/i386/strobe-1.04-2.i386.html*. RPM stands for Red Hat Package Manager, a utility that bundles the various files making up a Linux program and permits easy management of the resulting software. Other varieties of Linux may optionally support RPM as well.

Vincent LeVeque is a senior security engineer for Science Applications International Corporation (SAIC). He can be reached at *vleveque@earthlink.net*.

REFERENCES AND RELATED MATERIALS

OS/400 CL Reference V4R1 (SC41-5722-00)

Finding Flighty