

[print](#) | [close](#)

Print Program Adoption Supercharged!

[System iNEWS Magazine](#)

[Carsten Flensburg](#) [Dan Riehl](#) [Dan Riehl](#)

Carsten Flensburg

Wed, 06/01/2005 (All day)

[Click here to download code](#)

Program Adoption of Authority (PAA) is a technique that you can use to temporarily elevate users' authorities so that the users can perform functions that they would usually have insufficient authority to do. For example, you want the help desk folks to be able to reset a user's password. To do this sensitive function, the help desk personnel would need high authority levels (e.g., *ALLOBJ, *SECADM). But if you assign these high authority levels to the help desk staff's user profiles, these users then have full-time power to all your sensitive data, and this is dangerous to the system's overall integrity.

What you need to do instead is provide a small, limited-function program that such users can run to adopt the powerful authority that they need to do the password reset. When the adopting program runs to completion, the elevated level of authority is removed. This is what PAA does.

PAA is a useful function that you shouldn't shy away from. However, in the wrong hands, PAA can be the back door for obtaining high levels of authority without proper oversight and control.

In my work doing OS/400 vulnerability assessments, I've encountered many instances in which a program that adopts a high level of authority has been surreptitiously created on the system and used as a method to "back door" otherwise well-planned security. Using one of these adopting programs, a technical user can easily adopt a powerful user's authority, such as QSECOFR, and virtually become QSECOFR anytime he or she likes.

Creating These Adopting Programs

So how could I back door your system with PAA? Here's the source code for a simple CL program, MYPGM, which, when prepared properly, lets me run under QSECOFR authority anytime I like. But it takes more than just the source code.

```
PGM
GO MAIN
ENDPGM
```

Now that I have the source code, all I need is the opportunity to create the program and set up the adoption of QSEC OFR for this program.

So . . . I have a production emergency in the middle of the night and need to get the QSECOFR password to fix the problem. When I have the password, I sign on as QSECOFR and fix the production problem. Now, to get the back-door program, I run the following two commands:

```
CRTCLPGM  PGM(Library-Name/MYPGM)
          SRCFILE(Library-Name/QCLSRC)
```

```

SRCMBR (MYPGM)
USRPRF (*OWNER)
AUT (*EXCLUDE)

```

This command creates the CL program MYPGM. The USRPRF(*OWNER) parameter tells the system that when this program runs, it adopts the authority of the program's owner. In this case, because QSECOFR created the program, QSECOFR is the owner.

```

GRTOBJAUT OBJ (Library-Name/MYPGM)
          OBJTYPE (*PGM) USER (ME)
          AUT (*USE) REPLACE (*YES)

```

This command gives the user profile ME the authority to run the program anytime. And what does the program do? It adopts QSECOFR authority and sends me to the OS/400 main menu. From the main menu, I can do anything. *I have the power!*

From now on, in my day-to-day work, when I want to have QSECOFR authority, I no longer need the QSECOFR password. All I have to do is to call MYPGM. Now I can even set the QSECOFR password to anything I like.

Finding These Adopting Programs

Now that you understand how authority-adopting programs are created and the damage that someone who uses them wrongfully can cause, let's discuss how to find such programs on your system. To help you hunt down these potentially dangerous adopting programs, IBM has provided the CL command DSPPGMADP (Display Program Adopt).

DSPPGMADP requires that you specify the USERID of the user profile whose authority is adopted in the program. The command searches through all the libraries on the system to find programs that adopt QSECOFR (or whichever high-authority user profile you specify). Although the DSPPGMADP command is useful, it has a big downside. To find all the programs that adopt powerful special authorities, such as *ALLOBJ, you must run the command once for each user who has that powerful special authority. (For more information about DSPGMADP, visit publib.boulder.ibm.com/infocenter/iserics/v5r3/ic2924/info/cl/dsppgmadv.htm.)

Print Program Adoption Supercharged

So, the basic problem with IBM's DSPPGMADP is that it reports on only one USERID for each execution of the command. Often, dozens of powerful user profiles exist on a system. You'd need a listing of every powerful USERID, and you'd then need to run the DSPPGMADP command for each one.

Here's the format of IBM's DSPPGMADP command:

```
DSPPGMADP USRPRF (QSECOFR) OUTPUT (*PRINT)
```

[Figure 1](#) shows a snippet of a report from IBM's DSPPGMADP command. To overcome DSPPGMADP's shortcomings, Carsten Flensburg wrote the CL command PRTPGMADPS. This command provides more information than IBM's DSPPGMADP command, and instead of selecting a USERID to report on, you select the special authorities that you're looking for, such as *ALLOBJ or *SECADM. With this method, you can efficiently find all programs and service programs that adopt the special authorities specified. Here's an example of how you'd run the command:

```
PRTPGMADPS  PGMLIB (*ALL)
             SPCAUT (*ALLOBJ *SECADM)
             ORDER (*LIBOBJ)  SYSOBJ (*YES)
             JOBD (*USRPRF)  OUTQ (*CURRENT)
```

This command searches all libraries on the system for programs and service programs that adopt *ALLOBJ and *SECADM special authority. The listing order is by library and by object name within the library. IBM-created objects are included in the report. If the command is run interactively, it submits a batch job according to the JOBD and OUTQ parameters' values. [Figure 2](#) shows the PRTPGMADPS command prompt.

[Figure 3](#) explains the PRTPGMADPS command's main parameters. The SYSOBJ parameter lets you filter out of the report objects created by the USERID *IBM, which is used on many QSYS IBM-supplied objects. If you specify *NO, the objects created by *IBM are excluded from the report.

The JOBD parameter lets you direct the submitted job to use a particular job description, and the OUTQ parameter lets you specify the output queue to direct the report to.

The Report Is Cool!

The report that PRTPGM ADPS produces contains a list of all the programs and service programs — residing in the specified libraries — that adopt the special authorities that you specified. [Figure 4](#) shows a sample report, and you can see that the report lists, among other values, the user profile adopted and the program creator. Often, these values are different, meaning that someone used the CHGOBJOWN (Change Object Owner) command to manipulate the program.

What to Look for in the Report

When I perform vulnerability assessments, I run this report and pay particular attention to the programs in nonsystem libraries (e.g., QGPL) or user libraries (e.g., BOB). CL programs that adopt *ALLOBJ authority are a favorite back door. So I look for CLPs (i.e., Original Program Model — OPM — CL programs) and CLLEs (i.e., ILE CL programs). To investigate them fully, I use the IBM-supplied command RTVCLSRC (Retrieve CL Source) so that I can see exactly what the program is doing — regardless of what the source currently looks like.

Yes, back-door adopting-program objects can be written in any OS/400-supported language. So I do make sure that I know the purpose of the programs that adopt.

But How Else Do I Fix Things at 3:00 a.m.?

I must admit that in my younger days, I usually found a way to create an adopting program that would give me *ALLOBJ authority. When you get a support call at 3:00 a.m., all you want to do is fix the darn thing and try to go back to sleep. So, I'd simply call my special adopting program, fix the problem, and try to go back to sleep.

I'm older and wiser now, and so is the boss! In the ultra-regulated environment that we work in today (e.g., Sarbanes-Oxley, Gramm-Leach-Bliley Act, HIPAA), we need to be much more careful about unrestricted access. Do we need to be able to fix production problems? Of course. But we need to have an audit trail of everything done under these elevated authority levels. No one should be able to access and manipulate production programs and data with anonymity.

Using the system's security auditing functions, we can now track all activities of our powerful users. Learn how in *iSeries Security Reference Version 5* (SC41-5302 —

publib.boulder.ibm.com/infocenter/iseriess/v5r3/ic2924/info/rbap/sc415302.pdf) or in my article "[Common Sense Security Auditing](#)" (August 2004, article ID 18842 at iSeriesNetwork.com).

And Vendor-Supplied Software

I've noticed that most OS/400 software vendors, especially tool vendors, like to adopt powerful special authorities in their programs. You can ask your software vendor why all that authority needs to be adopted, but the answer you receive will often be unsatisfactory. Some things you just have to live with, I guess . . . hmmm.

Author Dan Riehl would like to thank Carsten Flensburg, who wrote the code for this article.

Dan Riehl is the director of services for The Powertech Group, an iSeries security technology company. Dan is also a technical editor for **iSeries NEWS** and the editor of the iSeries Network **Club Tech Systems Management Newsletter**. You can e-mail Dan at dan.riehl@Powertech.com.

Carsten Flensburg has been an iSeries programmer since 1992 and currently works as an iSeries programming team leader for a European vacation rental company called Novasol, which is a part of the U.S.-based Cendant Corporation. You can e-mail Carsten at flensburg@novasol.dk.

STAY ON TOP OF ISERIES SECURITY WITH THIS E-NEWSLETTER

Every other Wednesday, the iSeries Network *Club Tech iSeries Systems Management Newsletter* is distributed to thousands of readers' e-mail boxes. It's free! In each issue, Carsten Flensburg and I try our best to provide content that you can really sink your teeth into. Over the last several months, we've been dealing quite a bit with OS/400 security management, presenting tips and some great tools to help you stay on top of your systems' security. This article offers an enhanced version of a utility that we first presented to the iSeries Network ProVIP members who subscribe to the newsletter. If you haven't already subscribed, go to iSeriesNetwork.com/provipcenter/subscribe.cfm and sign up today!

— D.R.

Instructions for Obtaining and Using the Code

The RTVSPCAUT command was presented in the February 2, 2005, issue of the *Club Tech iSeries Systems Management Newsletter*. RTVSPCAUT is used in the initial CPP of the PRTPGMADPS command and is therefore bundled with this month's utility.

The following source code is included:

CBX930X — Command Definition source code
CBX9301 — CL CPP
CBX9302 — RPG IV program that does the work
CBX930H — Command Help Panel Group
CBX930V — Command Validity Checking Program

See each source member for compile instructions. See the Help Panel Group for more documentation.

To download all the code, go to iSeriesNetwork.com/code and choose June 2005 from the drop-down box on that page. You'll see this article's title and the links to download the code for the utility.

— *D.R.*

Source URL: <http://iprodeveloper.com/security/print-program-adoption-supercharged>