PowerTech
your security expert

WHITE PAPER

Managing Privileged Users on IBM i

By Robin Tatam

Bring up the topic of IBM Power Systems™ and IBM i and the subject of server viability and platform longevity invariably comes up. For years, analysts have predicted the demise of the platform whose heart beats in the chest of many of the world's largest organizations. Frustrated industry experts reject the notion that the server is outdated, citing highly scalable 64-bit Power-PC technology, class-leading reliability, and integrated security.

Much of the Power Systems controversy involves its "green screen" user interface. For years, IBM and a veritable army of third-party solution providers have sought to provide the ultimate answer to what many regard as the server's Achilles heel. A steady stream of products provide users with a modern, graphical user interface (GUI) to their IBM i applications.

Despite our love/hate relationship with the green screen and strides to enable a GUI, IBM i remains primarily a command-based operating system. As a result, the operating system is uniquely vulnerable when access to server commands is lax. For this reason, best practices and modern regulatory compliance standards mandate that IBM i server commands must be restricted to users on an as-needed basis to ensure server and application integrity.

This paper exposes some little-known risks associated with commands and provides recommendations on ways command access should be managed. It also highlights the need for commercial-grade security solutions designed specifically to assist administrators, auditors, and managers with this often-difficult task.

Defining "Command"

Simply stated, a command is an interface to a server or application function. IBM i commands simplify the invocation of a program much like a Microsoft Windows shortcut icon. A command is a made up of a command definition object (*CMD), which defines any parameters and command rules, and an associated command processing program (CPP), which performs the command's function.

IBM supplies more than 1,700 commands to control the server's operating system and licensed program products. In addition, an easy-to-use programming interface enables software developers to create their own commands. In fact, PowerTech includes numerous commands to provide a fast-path for power users to directly access functions in our security solutions. In a fifteen-year programming career, I personally wrote hundreds of commands to simplify and streamline

The PowerTech Group, Inc.
www.powertech.com • info@powertech.com

TEL USA: 253-872-7788
TOLL FREE: +1 800-915-7700
TEL UK: +44 (0)870 120 3148

Copyright 2013. PowerTech is a registered trademark of The PowerTech Group, Inc. AS/400 and System i are registered trademarks of IBM. All other product and company names are trademarks of their respective holders.

the user experience with my applications. This paper focuses primarily on IBM-supplied commands, but most of the concepts are equally applicable to user-written commands.

In comparison to the direct invocation of a program, a command has three primary benefits:

- Easy to remember
- Simple to invoke
- Validation of user information (parameters)

Let's look at a fictitious order report program, *myorders*, which requires the user to supply qualifying information—perhaps a customer number, a data range, and whether out-of-state orders should be included—before it can execute its programmed task.

Programmers typically design a screen interface to facilitate this type of information entry; however, this makes automation difficult as the user must be sitting at a workstation and take a menu option to enter the information.

Depending on how the application is programmed, the user may be able to supply the required information directly to the program by including parameters. The call to our orders report program may resemble the following:

```
CALL PGM(myorders) PARM('000010'
'09012011' '093114' '*NO')
```

This approach eliminates the automation restriction by enabling users and batch programs to bypass the entry screen. However, it's not something that we would want end users typing on a command line; it's far too complex and the formatting is prone to human error. This is a perfect scenario where a command provides ease of use and flexibility.

Unlike the simple *myorders* program, IBM's operating system programs often have dozens of parameters. Parameters may also have co-dependencies; for example, the RSTOBJ (restore object) command allows the user to designate that a save file (*SAVF) is to be used as the source of the restored objects. If the user designates *SAVF then they also must provide the name and library location of the save file.

Allowing a user to call an IBM i command-processing program directly could make the server vulnerable to abuse (not to mention how complex and cumbersome the CALL command would be). Instead, IBM prohibits users from invoking operating system programs directly and provides commands and application programming interfaces (APIs).

Which users can run commands?

Misconceptions are common regarding how a user is granted command privileges. The operating system does not allow control over whether users can access a command line, but only whether they can run commands on it. Contrary to popular belief, control is not a system- or profile-wide setting, but is determined by each command individually.

Every user's profile contains an attribute that designates if the user has limited capabilities. This attribute has three possible values: *YES, *NO, and *PARTIAL. The *NO and *PARTIAL values grant a user command privileges. A value of *YES works in conjunction with a corresponding attribute on each command definition object which indicates if the command is available to the user with limited capabilities.

The vast majority of IBM i commands are shipped with an *Allow Limited User* attribute value of *NO which prevents execution by a user with limited capabilities. This command attribute can be altered and should be audited occasionally to ensure that it has not been modified without permission, especially for powerful or sensitive commands. Auditing *CMD objects will generate a "ZC" (object change access) event if this command attribute is modified, although the log will not indicate which command parameter was altered. All audit entries should be monitored and reported on with dedicated auditing solutions.

The limit capability setting is only applicable to commands that are invoked from a system command line. Commands still can be run within a program and initiated from a menu. This provides the flexibility to permit users indirect access to system functions, such as *Work With Spooled Files* (WRKSPLF), without giving them command execution permission. Unfortunately, as discussed below, client-server interfaces do not always enforce the limit capability restriction.

Running a Command

The system command line represents the most common way to execute commands; however, IBM i contains numerous interfaces that allow commands to be issued:

- 5250 (green screen) command line on operating system screens
- Within a command language (CL) program
- Programming APIs (e.g. QCMDEXEC)
- Submitted to batch
- Non-traditional green screen interfaces (e.g. QSHELL)
- Client-server interfaces (e.g. FTP, REXEC)

Understanding each methodology is critical for any chance of exercising control over commands. For example, a little-known vulnerability exists with several client-server interfaces, including FTP, which allow host commands to be invoked by any user. While object authority and special authority requirements must always be satisfied, not every interface validates the user's *limit capability* attribute. This means that users could be obtaining command privileges without the consent of the security administrator, regardless of the 'allow limited user' settings.

IBM i Command Restrictions

Once a user has been granted (or has illicitly obtained) the ability to invoke commands, several criteria remain that must also be satisfied before a command runs:

Object Authority

IBM i evaluates a user's authority to all objects in the same way regardless of the object type. A user must have sufficient authority to the command definition object to run the command.

Many operating system commands are shipped with a public authority of *EXCLUDE. This limits access to profiles with *ALLOBJ special authority and profiles that have been granted private authority to the command object. Other commands are supplied with a public authority of *USE allowing virtually any user to easily satisfy the requirement. The *Print Public Authority* (PRTPUBAUT) command should be used to determine which commands do not have public authority of *EXCLUDE. Of course, a security administrator can change public authority to tighten or relax the object authority on any command.

If executing a command will impact another object, the user must have the necessary authority to the impacted object in addition to authority to the command itself. For example, the *Delete Program* (DLTPGM) command is designed to delete program objects. In addition to having authority to the command, a user must have Object Existence (*OBJEXIST) authority (or *ALLOBJ special authority) to the program object being deleted.

If best practices are followed, IBM i event auditing should be active and configured to detect authority failure events (*AUTFAIL). Unauthorized access attempts will log an "AF" (authority failure) event into the security audit journal, QAUDJRN. To ensure timely notification of these security events, QAUDJRN should be monitored proactively with an audit solution.

For more information on object authority, refer to the *IBM i Security Reference Manual* available online at www.ibm.com (keyword "IBM i Security Reference Manual").

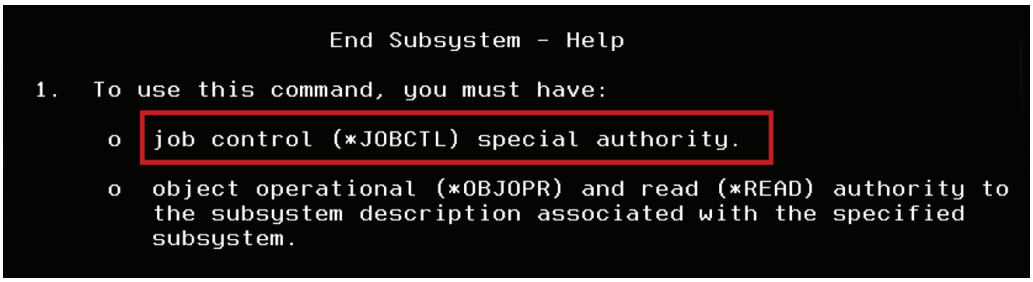
Special Authority

After obtaining the necessary object authority to the command and any impacted objects, the operating system determines if the user must also possess a particular special authority. There are eight administrative privileges—known as special authorities—that can be assigned to a user or inherited from any of the user's group profiles. Many commands mandate that the user must possess one or more of these special authorities.

IBM does not publish a list of commands requiring a particular special authority; however, each operating system command has detailed help text that outlines if there are additional requirements, including special authorities. An example of this documentation is shown in Figure 1.

Alternatively, the *Generate Command Documentation* (GENCMDDOC) command can generate HTML files in the Integrated File System (IFS) containing the help text.

FIGURE 1: ENDSBS IS AN EXAMPLE OF A COMMAND REQUIRING *JOBCTL SPECIAL AUTHORITY



Command Definition Restrictions

A command's definition object defines whether it can be executed interactively, in batch, or both. It also designates if the command only can be invoked from a program. These settings can be altered but there is usually a programmatic reason for them to remain set as supplied. The *Retrieve Job Attribute* (RTVJOBA) command, for example, only can be invoked within a CL program as it retrieves job information into variables for interrogation. This command would serve no purpose if invoked directly from a command line.

IMPLEMENTING A SOLUTION

Start with Perimeter Protection

Despite IBM's addition of exit point "hooks" in the operating system in the mid-1990s, PowerTech's *State of IBM i Security* study reports that the vast majority of organizations are still not using exit programs to monitor access through client-server interfaces. While not every interface provides data access or the ability to execute host commands, several do. In my professional opinion, this often represents the most significant configuration vulnerability.

Fortunately, this risk can be mitigated surprisingly easily by implementing a commercial exit program solution. An exit program should be assigned to each exit point so that it is invoked any time a transaction comes through the interface. A well-designed exit program fulfills the requirement of a software firewall to control and audit user transactions.

Security administrators should closely monitor traffic coming through these client-server interfaces and strongly consider restricting the ability to execute host commands. The ideal exit program solution can do this selectively, still permitting legitimate commands to be executed and audited. Legacy command execution restrictions are easily enforced once a user is not able to circumvent the limited capability restriction.

For more information on exit points and exit programs, refer to *The Truth Abut Exit Points*, an exclusive white paper available at www.powertech.com.

Reduce the Number of Privileged Users

Security best practices—as well as many modern regulatory standards—call for a restriction on the number of privileged user accounts. This mandate encompasses users that can access critical server functions as well as those that can access application data outside of an approved application. The goal of restriction is to prevent a user from having the ability to perform tasks that are outside of the responsibilities of that job role.

My definition of a privileged user is one that carries command privileges in conjunction with one or more of the following:

- Any of the eight Special Authorities
- Private authority to critical objects
- A server with permissive public authority

As a reminder, commands may be executed through client-server interfaces even when the user has limited capabilities. In addition, data access through these interfaces does not require command privileges.

One of the most frequently cited audit issues on servers running IBM i is that there are often far too many user accounts with unjustified privileges. Each year, the PowerTech security study corroborates this

with a review of special authority assignments and public authority settings. Fortunately, role-based access control (RBAC) initiatives are becoming commonplace and this helps identify profiles with excess privileges so that they can be permanently altered to a more appropriate configuration.

Of course, there are times when programmers, administrators, and even end users have a legitimate requirement to access restricted areas of the server and perform tasks that might require higher privileges than the corporate security policy allows. This can be accomplished while still retaining control and visibility to the user's actions.

Adopted Authority

Restricted tasks can be performed using *adopted authority*. Adoption is a programming-based technique that enables a program to run with more authority than the user who is running it. This temporary elevation of authority is best suited to performing very specific tasks, such as a enabling a disabled account. It solves the dilemma of granting a help desk employee *SECADM and *ALLOBJ special authorities in order to use the *Change User Profile* (CHGUSRPRF) command.

Care must be taken to ensure that the adopting program does not provide an interface to allow tasks beyond the program's intended function. For example, the adopting program should not present the user with a command line that honors the adopted authority, otherwise the user may use the increased authority to run unauthorized functions.

Auditors may require an audit trail of programs that elevate a user's authority via adoption. This can be accomplished by designating *PGMADP in the IBM i event audit configuration. Before you take this approach, check whether an application utilizes adoption as a primary authorization technique, as the setting will generate an audit entry every time an adopting program is invoked.

Profile Switching

Profile switching is the modern solution for the temporary assignment of privileged access. Also an operating system capability, switching carries several advantages over legacy authority adoption:

- User authority also can be reduced
- Effective for IFS access
- Can be utilized outside of an application program

Switching enables a user to inherit all of the capabilities of the target profile, including command line privileges. Unlike adoption, which can only maintain or increase a user's authority, switching can also *reduce* authority—useful when starting an interactive SQL or Query session. Powerful users such as programmers and operators can now be configured without privileges and then invoke a profile switch when a restricted task needs to be performed. IBM i APIs facilitate the seamless switching process—in fact, the operating system uses this technique for many of its own TCP/IP services.

Profile management software is ideal for controlling powerful users while still supporting the IT requirement for privileged account access. Using modern profile-switching APIs, the ideal solution facilitates a user's acquisition of additional privileges as-needed while satisfying the monitoring provisions of modern regulatory standards.

Restrict and Monitor Command Usage

IBM i is the primary enforcer of control over command execution. Unfortunately, there is no way to designate conditional criteria. Once a user has met the necessary criteria for a particular command, the operating system does nothing else to ensure an activity is reasonable. For example, an operator who has been granted *JOBCTL special authority to end the server's subsystem during a weekend maintenance window can also accidentally perform that same task in the middle of the production day.

For many organizations, using software to manage command line access has quickly become an integral part of their command controls. A powerful solution supplements traditional authority with conditional criteria and action-based responses. The ideal solution enables authorized users to designate complex conditions that are evaluated prior to IBM i processing the command.

A partial list of available conditions may include:

- Command parameter evaluation
- User is on a named authorization list
- Day of week
- System date and time
- User name
- If a named program exists in the invocation stack

If a designated condition is evaluated to be true, then an action can be performed. A partial list of actions may include:

- Prevent the command
- Continue with the command as-is
- Run an alternate command (in addition to or in replacement of the requested command)
- Add/remove/override a command parameter value
- Send a notification

For the first time in the history of the IBM i (OS400, i5/OS) operating system, administrators can now designate *when* and *how* a command can be used. It is easy to prevent the system operator from accidentally powering the server down during the day while permitting the same user to run it during a scheduled maintenance window—perhaps with an authorization list to designate "super" administrators who retain full control. You can also force programmers to use a switch profile in order to track and control development activities, despite the fact that the operating system may grant these users unrestricted use of the compilers.

A command monitoring program must evaluate conditions and perform actions *before* the command is executed. This way control can be imposed on users who carry the (normally) unrestricted power of *ALLOBJ special authority! Imagine preventing a security officer from turning off the system audit functions, or sending an alert notification the instant that a user profile is created outside of the PowerTech PowerAdmin user propagation facility.

Auditing User Actions

Talk to an auditor or dig into any regulatory standard and you most likely will discover that auditing the activities of powerful users is a primary initiative. In fact, audit standards pertaining to privileged access are often as concerned with the visibility of actions as they are with preventing unauthorized activities.

IBM i contains powerful functionality to track system and user events. To learn more, visit *Security Auditing in the Real World*, a free download at www.powertech.com. This exclusive paper outlines how to set up auditing at the system, user, and object level. It also contains valuable information regarding methods to interrogate the resulting audit data.

In the context of this document, we are primarily interested in tracking the command activities of particular individuals. In order to support the separation of duties between the role of security administrator and auditor, the Change User Audit (CHGUSRAUD) command is used instead of the normal profile configuration commands. Assigning the value of *CMD instructs IBM i to record commands issued by the user or through a program that they are running. The *CMD value is available only with specific users due to its potential to generate an excessive volume of audit entries.

When a command is executed by an audited user, a "CD" entry is written to the event repository (QAUDJRN). This provides irrefutable proof that the command was run and by whom.

The challenge with auditing—regardless of the types of events—is that IBM provides only rudimentary commands to extract and analyze the audit data. Responsibility remains with the organization to locate and process the entries, in addition to reacting to the use of an unauthorized command. Some solutions can automatically configure all audit settings and simplify reporting over command events. Reporting functions should allow each individual's activities to easily be identified.

The Danger of Invisible Commands

There is nothing in IBM i that truly constitutes an "invisible" command; however, there are numerous commands that will transport the user into an environment that cannot be audited using the command audit facility discussed previously. Auditing of these "invisible" commands must be addressed to provide a comprehensive trail of a user's command activities.

Examples of potentially dangerous and hard-to-audit commands include:

- Start Data File Utility (STRDFU / UPDDTA)
- Start Interactive SQL Session (STRSQL)
- Start QShell Session (STRQSH)
- Work Link (WRKLNK)
- Work With Query/400 (WRKQRY)
- Call Program (CALL)
- Start Program Development Manager (STRPDM)
- Start Source Entry Utility (STRSEU)
- Start System Service Tools (STRSST)

These commands are hard to audit because they initiate an environment that allows a user to perform system, programming, or database tasks without generating a comprehensive audit trail. Database changes can be audited by journaling the physical files, although it's often difficult to tie the change to the action that caused it. For example, let's consider the following SQL statement:

```
Update CUSTOMER set ARBALNC = 0
where ARBALNC > 0
```

This simple instruction will instantly zeroize the ARBALNC field on every record. If the CUSTOMER file was being journaled, then the journal receiver would contain an entry for each record that was updated—potentially tens of thousands of entries—but it would be virtually impossible to identify what process instigated the change.

A common request from IT auditors is to know *exactly* what tasks programmers are performing on production systems. Programmers are usually the most experienced and powerful users on the server and represent a very real threat potential. Auditing program objects will indicate that an object was opened with the potential for change, but providing a comprehensive log of program source code changes is difficult and this is unacceptable in an audit.

Methodologies exist to trace SQL statements but do nothing to audit any of the environments identified above. To mitigate this risk, your solution must contain a screen capture capability. With this facility, when the user initiates a swap, a virtual "camera" can be activated to record all interactive activity. Having an audit trail that includes all 5250 screens, keyboard entries, and which key was pressed to process the screen provides a comprehensive, yet easy-to-read trace of a user's activities.

Ideally, the screen capture function should enable an administrator to view privileged users' screens in near real-time. It should also allow post-session playback to view user activities after they have completed. This can be especially useful for monitoring off-shift users or non-employees such as consultants and vendors—an area of grave concern for auditors. In addition, some screen capture features on the market can bundle the recorded screens into a searchable PDF and automatically distribute them as an email attachment upon conclusion of the swap activity.

Next Steps

Managing privileged users on the IBM i server requires a coordinated approach to be effective. Legacy controls can break down quickly in the modern world, where access comes from PC-based interfaces as well as the traditional command line. The biggest mistake an organization can make is to do nothing.

When embarking on a privileged account management project, an organization should:

Review

Interrogate the currently assigned privileges to determine existing capabilities.

Define roles based on server and applications.

Identify privileges to be associated with each user role.

Cleanse

Remove unnecessary privileges from users.

Establish group profiles for roles.

Assign users to roles.

Enhance

Implement security applications to increase control and visibility of user activities. Solutions should:

- Control user on- and off-boarding to guarantee consistency in the deployment of user roles.
- Secure and audit client-based command lines and database access.
- Facilitate temporary assignment of privileges via profile swaps, tracking of legacy command line activities, and recording of user screens.
- Provide notification and conditional control over commands.

To be truly effective, auditing should be continuous. Checklists should include verification of appropriate user privileges and a review of the audit log of profile swaps. If swap activities are frequent, consider random spot-checks in addition to reviewing activities with partial or missing information (e.g. incorrect ticket numbers or weak explanations).

In Summary

As long as IBM i remains a command-centric operating system, controlling and monitoring the use of commands will be a critical security requirement for all organizations. Utilizing object-level security and command line restrictions establishes a solid foundation upon which other controls can be built. When best practices in IBM i security configuration, monitoring, and reporting are not possible—or not flexible enough—quality third-party solutions should be assessed to enhance the features and simplify the effort.

PowerTech's security solutions run in some of the largest companies on the planet. Market-proven, they enable these organizations to accomplish compliance faster and more easily, thereby saving costs while reducing risk. Organizations that (currently) have no regulatory requirements can still benefit from the enhanced security experienced through their deployment.

When synergized by a well-configured IBM i security infrastructure, PowerTech solutions elevate your Power Systems server running IBM i to a world-class security environment for managing access by privileged users.

About the Author

Robin Tatam is the Director of Security Technologies for PowerTech, a leading provider of security solutions for IBM i servers. A frequent speaker on security topics, Robin is a Security subject matter expert for COMMON, the world's largest IBM i user group, and is co-author of the IBM RedBook "System i® Security: Protecting i5/OS Data with Encryption." *Robin can be reached by email at* ***robin.tatam@powertech.com.***

APPENDIX A

Powerful commands

A selection of powerful commands and their associated attributes are outlined in Table 1. This is not an exhaustive list due to variations in licensed program products as well as the sheer number of IBM i commands. It is designed to identify commonly used commands that could be considered invasive. More information about the function of each command can be found in the IBM Information Center.

TABLE 1: POWERFUL COMMANDS

Command	Description	Allow Limited Capability Users	Public Authority	Special Authority Requirement	Other Reqs
ADDEXITPGM	Add / Remove Exit Program	No	*EXCLUDE	N/A	No
ADDJOBSCDE	Add Job Schedule Entry	No	*USE	N/A	Yes
CHGDDMTCPA	Change DDM Attributes	No	*USE	*IOSYSCFG	No
CHGFCNUSG	Change Function Usage	No	*USE	*SECADM	No
CHGNETA	Change Network Attributes	No	*EXCLUDE	*IOSYSCFG (+ *ALLOBJ for some values)	No
CHGSVRAUTE	Change Serve Authentication Entries	No	*USE	*SECADM	Yes
CHGSYSVAL	Change System Value	No	*EXCLUDE	Dependent on values	No
CHGUSRPRF	Change User Profile	No	*USE	*SECADM	Yes
CRTLIB	Create Library	No	*USE	N/A	No
CRTxxxPGM	Create Programs	No	*USE	N/A	Yes

Requirements might include object-level authority to an object impacted by the command. Information on command requirements can be found in the help text for the command.

TABLE 1: POWERFUL COMMANDS (CONTINUED)

Command	Description	Allow Limited Capability Users	Public Authority	Special Authority Requirement	Other Reqs
DLTLIB	Delete Library	No	*USE	N/A	Yes
DLTxxx	Delete Object	No	*USE	N/A	Yes
ENDSBS	End Subsystem	No	*USE	*JOBCTL	Yes
ENDTCPSVR	End TCP Server	No	*EXCLUDE	N/A	No
PWRDWNSYS	Power Down System	No	*EXCLUDE	*JOBCTL	Yes
RMVEXITPGM	Add / Remove Exit Program	No	*EXCLUDE	N/A	No
RSTLIB	Restore Library	No	*EXCLUDE	*SAVSYS or private authority	Yes
RSTOBJ	Restore Object	No	*EXCLUDE	*SAVSYS or private authority	Yes
SAVLIB	Save Library	No	*USE	*SAVSYS or private authority	Yes
SAVOBJ	Save Object	No	*USE	*SAVSYS or private authority	Yes
STRSST	Start System Service Tools	No	*EXCLUDE	*SERVICE	No
WRKLNK	Work Link	No	*USE	N/A	Yes
WRKSPLF	Work with Spooled Files	No	*USE	N/A	Yes
WRKJOSBSCDE	Work With Job Schedule Entries	No	*USE	N/A	Yes