Payoff

AS/400 systems offer a wide array of powerful mechanisms for information security and auditing. The security manager must be able to use these controls to protect data without sacrificing system efficiency. This article presents a program for building effective controls that minimize the resources needed to manage the security function.

Problems Addressed

In an IBM Application System 400 (AS/400) environment, the security manager must establish security controls that adequately protect data without impeding system throughput and ease of use. The security manager and staff must then monitor the system to ensure that all necessary controls have been activated and are functioning as expected.

This article presents a program for establishing and maintaining an effective level of control in AS/400 systems, regardless of the type of system configuration employed. The key control options are discussed and recommendations are presented for setting control parameters that strike the proper balance between information protection and system availability and processing efficiency. Of course, it is not possible to address every possible set of business requirements—some organizations have more stringent security requirements than others. The article assumes a business environment in which there is a relatively even mix of sensitive and nonsensitive business data, and provides guidance as to the appropriate controls for specific security requirements.

The AS/400 provides a wide array of security mechanisms that may seem bewildering to a security specialist who is not familiar with the AS/400 architecture. As with any security implementation, it is important that the security manager first develop a security plan for the design and implementation of controls. This is not just to ensure that a critical control feature has not been overlooked. An effective plan should also ensure that the costs associated with the proposed security program, including system overhead and the human resources required to manage the security operation, do not exceed the benefits. (Put differently, the goal should be to provide enough protection so that the cost of stealing the data exceeds its value.) A poorly designed security implementation can often result in excessive controls, which in turn make security management a burden.

This article provides a structured plan for establishing the necessary user and group profiles, object authorities, authorization lists, and library controls to minimize the resources needed to manage the security function. Of course, given the often rapid changes in computing environments, any plan designed today will need to be periodically reviewed and updated to satisfy new requirements.

The following section provides an overview of typical AS/400 configurations as well as a description of the basic system architecture. The remaining sections of the article provide details on establishing security controls for each system component.

In October 1995, the AS/400 formally received its first C2 security rating from the United States Department of Defense. This original C2 security rating is for V2R3 of OS/400, Source Entry Utility, Query/400, SAA Structured Query Language/400, and Common Cryptographic Architecture Services/400. The C2 security rating was awarded after a rigorous, multi-year period of evaluation. The AS/400 is the first system to achieve a C2 security rating for a system (hardware and operating system) with an integrated, full-function database.
Object-Based System Architecture

On the AS/400 system, data in the form of either files or programs is stored in storage units referred to as objects. Exhibit 1 illustrates that all objects consist of two basic types of information: common attributes that specify such information as object name, type, and ownership; and object contents that, for example, consist of data records in the case of files and executable instructions in the case of programs. The common attributes are described in detail in Exhibit 2.

**AS/400 Object Structure**

**Common Attributes of Objects**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Name</td>
<td>The user-assigned name used to locate the object. Objects of a different type can have the same name. For example, a file can be named PAYROLL and the program that processes the file can also be named PAYROLL.</td>
</tr>
<tr>
<td>Object Library</td>
<td>The user-assigned directory name used to group objects. Objects are located by library and object name. Objects of the same type can have the same name if they exist in different libraries. For example, two files can be named PAYROLL if one is in the PRODUCTION library and the other in the TEXT library. The contents of these two files do not need to be the same. Each job in the system has a library search list called a library list; this is similar to path specifications in other systems. The library search list is used if no library name is specified to locate an object. It should be noted that the library object type is used to store the name of objects and the virtual address used to locate the object.</td>
</tr>
<tr>
<td>Object Type</td>
<td>There are more than 80 different object types, which are assigned according to the type of object contents (e.g., files, programs, and queues).</td>
</tr>
<tr>
<td>Object Owner</td>
<td>The user responsible for authorizing others to use the object. The object owner is usually the person who created the object.</td>
</tr>
<tr>
<td>Owner Access to Object</td>
<td>The owner has by default *ALL access to an object, but may revoke some privileges (e.g., to prevent accidental deletion of the object). The owner can always grant authority to the object.</td>
</tr>
<tr>
<td>Public Access to Object</td>
<td>By default, users not otherwise authorized to access the object are assigned *PUBLIC access.</td>
</tr>
<tr>
<td>Authorized Users</td>
<td>Users specifically authorized to access an object are stored in the user profile for each user rather than in a centralized data base. For further information, see the section on user profiles.</td>
</tr>
</tbody>
</table>
Primary Group Profile  
A user profile other than the owner, which has access to an object. Only user profiles with a group identification number (gid) may be the primary group for an object.

Primary Group's Access  
Primary group authority is stored with the object and may provide better performance than private authority granted to a group profile. Primary group authority is not considered a private authority because the authority is stored with the object and not the user profile.

Authorization List (optional)  
Users can also be authorized to access an object using an authorization list. Authorization lists are discussed in detail in the section on object authority.

Object Audit  
The auditing of objects can involve no audit, logging of modifications, or logging of all references. The object auditing may specify that auditing be specified in user profile so that the object is audited for individual users.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Name</td>
<td>The user-assigned name used to locate the object. Objects of a different type can have the same name. For example, a file can be named PAYROLL and the program that processes the file can also be named PAYROLL.</td>
</tr>
<tr>
<td>Object Library</td>
<td>The user-assigned directory name used to group objects. Objects are located by library and object name. Objects of the same type can have the same name if they exist in different libraries. For example, two files can be named PAYROLL if one is in the PRODUCTION library and the other in the TEXT library. The contents of these two files do not need to be the same. Each job in the system has a library search list called a library list; this is similar to path specifications in other systems. The library search list is used if no library name is specified to locate an object. It should be noted that the library object type is used to store the name of objects and the virtual address used to locate the object.</td>
</tr>
<tr>
<td>Object Type</td>
<td>There are more than 80 different object types, which are assigned according to the type of object contents (e.g., files, programs, and queues).</td>
</tr>
<tr>
<td>Object Owner</td>
<td>The user responsible for authorizing others to use the object. The object owner is usually the person who created the object.</td>
</tr>
<tr>
<td>Owner Access to Object</td>
<td>The owner has by default *ALL access to an object, but may revoke some privileges (e.g., to prevent accidental deletion of the object). The owner can always grant authority to the object.</td>
</tr>
<tr>
<td>Public Access to Object</td>
<td>By default, users not otherwise authorized to access the object are assigned *PUBLIC access. Users specifically authorized to access an object are stored in the user profile for each user rather than in a centralized data base. For further information, see the section on user profiles.</td>
</tr>
<tr>
<td>Authorized Users</td>
<td></td>
</tr>
<tr>
<td>Primary Group Profile</td>
<td>A user profile other than the owner, which has access to an object. Only user profiles with a group identification number (gid) may be the primary group for an object.</td>
</tr>
</tbody>
</table>
Objects with similar contents are assigned the same object type—for example, files, programs, and queues. Because security is defined as a common attribute, the same security interface can be used for all objects, regardless of type.

Object implementation provides improved data integrity by preventing invalid operations for a given object type. For example, the call operation is defined as valid for program objects; open, read, and write operations are valid for file objects. Any attempt to call a file or open a program will be rejected. Such strong controls are a significant improvement over systems in which all data is represented as system files. The object typing of AS/400 programs, for example, virtually eliminates the potential for a viral infection of an AS/400 program, because the program object type does not permit one program to modify another.

**User Profiles**

A user profile is the object used by the system to identify the access and processing capabilities of the user. Profiles are used to identify users in all audit records; therefore, the security administrator should create a user profile for every system user. (Several administrators can be designated to assist with the enrollment and removal of users.)

The user profile contains information for security applications as well as general purposes. The following general information is used to adapt the system to individual users:

- The user assistance level, which specifies the amount of help text that appears on system screens. Levels of *BASIC, *INTERMEDIATE, and *ADVANCED can be specified.
- The current library, which indicates the library name used when new objects are created by the user.
- The previously accessed library and file for each system utility.
- The accounting code used to track the use of system resources for company projects.
- The job description, which establishes the operational environment for the user's processing jobs.

Other general options can be used to control type-ahead capabilities, prompt messages, the level of detail on system screens, and the printing of the user name on output.

The user profile also contains the security-related fields listed in Exhibit 3. In general, these fields control the user's ability to access, add, modify, and delete objects and to use certain system services.

**Security-Related Fields in the User Profile**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRCLS</td>
<td>The User Class field identifies the role of the user in the system. The choices from highest to lowest level of responsibility are:</td>
</tr>
<tr>
<td>User Class</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>*SECOFR</td>
<td>The security officer has unrestricted access to all objects. Use of this user class should be limited because there is no protection from accidental deletion of objects. The recommended procedure is to provide two user profiles for individuals that are security officers. The nonsecurity officer profile should be used for most system access and the security officer profile should be used only when required.</td>
</tr>
<tr>
<td>*SECADM</td>
<td>The security administrator is authorized to enroll and remove system users. A different security administrator should be assigned to manage the user profiles for each major functional area. The security administrator typically creates and owns the user profiles and, therefore, is authorized to access them. Profiles owned by other administrators cannot be accessed.</td>
</tr>
<tr>
<td>*PGMR</td>
<td>The programmer is authorized to add applications to the system.</td>
</tr>
<tr>
<td>*OPERATOR</td>
<td>The operator is permitted to manage system work and perform backup operations.</td>
</tr>
<tr>
<td>*USER</td>
<td>The user identifies general system users who have no special privileges.</td>
</tr>
<tr>
<td>SPCAUT</td>
<td>The Special Authority field identifies additional user privileges. If no privileges are specified explicitly for this field, the system defaults to values based on the selected User Class authorities. (To simplify the enrollment process, it is suggested that only User Class parameters be specified.) The following special authorities can be granted:</td>
</tr>
<tr>
<td>*ALLOBJ</td>
<td>The user has unrestricted access to all system objects (i.e., *All authority).</td>
</tr>
<tr>
<td>*AUDIT</td>
<td>The user is allowed to change the event and object audit related options in system values, objects, and user profiles.</td>
</tr>
<tr>
<td>*SECADM</td>
<td>The user is allowed to enroll and remove users from the system. Users granted both *ALLOBJ and *SECADM privileges are considered to be security officers. These users can change the system values, objects, and network attributes that control system security.</td>
</tr>
<tr>
<td>*SERVICE</td>
<td>The user is allowed to use system service tools. Service tools should be used only by trusted and knowledgeable users, because these tools can be used to circumvent system security controls. For example, improper use of service tools to display or alter an object may result in system failure. A communication line trace might also be used to view line transactions, including the sign-on information of other users.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>*SPLCTL</td>
<td>The user can access any spool files on the system. *SPLCTL provides the same authority for spool files that *ALLOBJ provides for objects. (Spool files are not defined as external objects.)</td>
</tr>
<tr>
<td>*SAVSYS</td>
<td>The user can save and restore objects that he or she is not normally authorized to use. This privilege is frequently given to individuals responsible for backing up the system.</td>
</tr>
<tr>
<td>*JOBCTL</td>
<td>The user can manage the jobs of other users. This authority is given to system operators.</td>
</tr>
<tr>
<td>USRCLS</td>
<td>*SECOFR The security officer has unrestricted access to all objects. Use of this user class should be limited because there is no protection from accidental deletion of objects. The recommended procedure is to provide two user profiles for individuals that are security officers. The nonsecurity officer profile should be used for most system access and the security officer profile should be used only when required.</td>
</tr>
<tr>
<td>SPCAUT</td>
<td>*IOSYSCFG The user is allowed to change how the system is configured, such as adding or removing communications configuration information. Most commands for configuring communications require *IOSYSCFG special authority.</td>
</tr>
<tr>
<td>GRPPRF</td>
<td>The Group Profile field allows users to share the authority of another profile. The section on group profiles describes this option in greater detail.</td>
</tr>
<tr>
<td>SUPGRPPRF</td>
<td>The Supplemental Group Profile field specifies up to 15 names of group profiles to be used with this user profile. These profiles, in addition to the profile specified in the GRPPRF parameter, are used to give the user access to objects.</td>
</tr>
<tr>
<td>INLMENU or INLPGM</td>
<td>The Initial Menu field or Initial Program field is used to determine the screen a user first sees when signing on. These parameters are frequently used when it is desired to limit users to selecting choices from a menu. Limiting users to menu choices requires a system security level level of 20 or higher.</td>
</tr>
<tr>
<td>LMTCBP</td>
<td>The Limit Capability field indicates that users are restricted from entering commands and changing the initial program on the sign-on menu. If users are limited to selecting options from a menu, the system default *NO mshould be changed to *YES.</td>
</tr>
</tbody>
</table>

**USRCLS**

The User Class field identifies the role of the user in the system. The choices from highest to lowest level of responsibility are:

- ***SECOFR** The security officer has unrestricted access to all objects. Use of this user class should be limited because there is no protection from accidental deletion of objects. The recommended procedure is to provide two user profiles for individuals that are security officers. The nonsecurity officer profile should be used for most system access and the security officer profile should be used only when required.
**Group Profiles**

Group profiles are used to define the authorization for a group of related users in a single profile rather than repeating the same authorization for each user profile. The authorization to objects and the special authority from the group profile are shared by all members of the group. Group profiles are widely used in AS/400 installations because their use simplifies security management and reduces the number of authorizations that must be stored. If the number of stored authorizations is reduced, less time is needed to back up the system.

Assigning users their own profiles is a good security practice. Users can change their password periodically without having to inform others about the change; and more important, it is only possible to identify and monitor individual user activity on the system when signing on to the system.

Where necessary, a user profile can be established with more or less authority than the groups to which it belongs. In such cases, the object authority granted in the user profile supersedes the authority specified in the associated group profiles. However, any special authority established in the user profile does not supersed the special authority in the group profile. Rather, the special authorities defined in both the user and group profiles are combined additively to determine the effective special authority.

A group profile is not a distinct AS/400 object type. A group profile is simply a special type of user profile. It becomes a group profile when one of the following occurs:

- Another profile designates it as a group profile.
- The user profile is assigned a group identification number (GID).

A designated group profile cannot be a member of another group. Individuals may be members of up to 16 group profiles.

**Managing User and Group Profiles**

Control Language (CL) commands are used to manage the assignment of user and group profiles as well as object authorities and authorization lists. These commands are described in Appendix A, which appears at the end of this article.

The same CL commands used to manage individual user profiles are also used to manage group profiles. The CRTUSRPRF command is used to create the user profile and the group profile. In naming group profiles, it is recommended that a consistent naming convention be used to distinguish groups so that they may be more readily identified on lists of authorized users. For example, the group name might begin with the characters DPT or GRP, signifying that group members belong to a department within the organization.

To create a group profile named DPTSALES, the following command could be used:

```
CRTUSRPRF USER(DPTSALES) PASSWORD(*NONE) SPCAUT(*JOBCTL)
```

A password of *NONE is recommended to prevent users from signing on under the group profile.

When individual user profiles are created, they can be assigned membership in a group by using the GRPPRF command parameter. For example, to create two users as members of the group profile, DPTSALES, the following CRTUSRPRF commands can be used:

```
CRTUSRPRF USER(GRANT) PASSWORD(--) GRPPRF(DPTSALES) + OWNER(*GRPPRF)
```
Both users Grant and Evans are members of the DPTSALES group and share the object and special authorities defined for this group. However, Evans is also granted *SAVSYS special authority. Because the option OWNER(*GRPPRF) is specified, any objects created by these users will be owned by the DPTSALES group profile. This option is particularly useful in cases where members of a group work on shared projects, because objects created by one group member immediately become available to all members of the group.

**Object Authority**

An object authority specifies the type of operation that can be performed on a given object. Object authorities are granted to either a user profile or *PUBLIC. Exhibit 4 lists the authorities that can be granted. Because there is no hierarchical relationship among authorities, it is possible, for example, to grant *ADD authority without *READ authority. This arrangement is useful in assigning authority to the message queue of another user; *ADD authority allows messages to be sent to the user, but without *READ authority the user's messages cannot be read by others.

AS/400 also supports a set of object authorities that permit the use of a single term to grant a combination of the individual object authorities listed in Exhibit 4. For example, the *CHANGE authority can be used to grant authorities to display, read, add, update, and delete an object. The combined object authorities, including the *EXCLUDE authority already introduced, are shown in Exhibit 5. Use of these combined terms is recommended.

**Object Authorities**

<table>
<thead>
<tr>
<th>Authority</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*ADD</td>
<td>Add authority is required to insert new entries into an object.</td>
</tr>
<tr>
<td>*AUTLMGT</td>
<td>Authorization List Management authority is required to add or remove users from an authorization list.</td>
</tr>
<tr>
<td>*DLT</td>
<td>Delete authority is required to remove existing entries from an object. Delete authority allows only deletion of individual data entries, not the entire object.</td>
</tr>
<tr>
<td>*EXCLUDE</td>
<td>Exclude authority prevents access to an object. If a user has no authority to an object, the user may still be able to access the object using the *PUBLIC authority. However, if the user is granted *EXCLUDE authority, all access is denied.</td>
</tr>
<tr>
<td>*EXECUTE</td>
<td>Execute authority is required to run a program, service program, or SQL package or locate an object in a library or a directory.</td>
</tr>
<tr>
<td>*OBJALTER</td>
<td>Object alter authority is required to add, clear, initialize and reorganize members of the database files; to alter and add attributes of database files; and to add and remove triggers.</td>
</tr>
<tr>
<td>*OBJREF</td>
<td>Object reference authority is required to specify a database file as the parent file in a referential constraint.</td>
</tr>
<tr>
<td>Authority</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>*OBJEXIST</td>
<td>Object Existence authority is required to delete an object or to save an object to backup media.</td>
</tr>
<tr>
<td>*OBJMGT</td>
<td>Object Management authority is required to authorize other users to an object or to move or rename an object.</td>
</tr>
<tr>
<td>*OBJOPER</td>
<td>Object Operational authority is required to display the description of an object; it is also required to open a file.</td>
</tr>
<tr>
<td>*READ</td>
<td>Read authority is required to retrieve information from an object.</td>
</tr>
<tr>
<td>*UPD</td>
<td>Update authority is required to modify existing entries in an object.</td>
</tr>
<tr>
<td>*ADD</td>
<td>Add authority is required to insert new entries into an object.</td>
</tr>
<tr>
<td>*AUTLMGT</td>
<td>Authorization List Management authority is required to add or remove users from an authorization list.</td>
</tr>
<tr>
<td>*DLT</td>
<td>Delete authority is required to remove existing entries from an object. Delete authority allows only deletion of individual data entries, not the entire object.</td>
</tr>
<tr>
<td>*EXCLUDE</td>
<td>Exclude authority prevents access to an object. If a user has no authority to an object, the user may still be able to access the object using the *PUBLIC authority. However, if the user is granted *EXCLUDE authority, all access is denied.</td>
</tr>
<tr>
<td>*EXECUTE</td>
<td>Execute authority is required to run a program, service program, or SQL package or locate an object in a library or a directory.</td>
</tr>
<tr>
<td>*OBJALTER</td>
<td>Object alter authority is required to add, clear, initialize and reorganize members of the database files; to alter and add attributes of database files; and to add and remove triggers.</td>
</tr>
<tr>
<td>*OBJREF</td>
<td>Object reference authority is required to specify a database file as the parent file in a referential constraint.</td>
</tr>
<tr>
<td>*OBJEXIST</td>
<td>Object Existence authority is required to delete an object or to save an object to backup media.</td>
</tr>
<tr>
<td>*OBJMGT</td>
<td>Object Management authority is required to authorize other users to an object or to move or rename an object.</td>
</tr>
<tr>
<td>*OBJOPER</td>
<td>Object Operational authority is required to display the description of an object; it is also required to open a file.</td>
</tr>
<tr>
<td>*READ</td>
<td>Read authority is required to retrieve information from an object.</td>
</tr>
<tr>
<td>*UPD</td>
<td>Update authority is required to modify existing entries in an object.</td>
</tr>
</tbody>
</table>
Combined Object Authorities

<table>
<thead>
<tr>
<th>OBJ</th>
<th>OPR</th>
<th>READ</th>
<th>*OBJ</th>
<th>EXE</th>
<th>CUTE</th>
<th>*ADD</th>
<th>*UPD</th>
<th>MGT*</th>
<th>*OBJ</th>
<th>EXIST</th>
<th>OBJ</th>
<th>REF*</th>
<th>OBJ</th>
<th>ALTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*EXELUDE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*USE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*CHANGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*ALL</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The CL commands listed under the Object Authority Commands heading in Appendix B are used to create and maintain object authorities. For example, the GRTOBJAUT command is used to grant a user profile authority to an object, while the RVKOBJAUT command revokes such authority. The full command statement must include the object name and type and the profile name. For example, to give Evans the authority to change the SAMPLE program, the following GRTOBJAUT command can be entered:

```
GRTOBJAUT OBJ(SAMPLE) OBJTYPE(*PGM) USER(EVANS) + AUT(*CHANGE)
```

AS/400 Systemwide Security Options

The systems manager specifies system values and network attributes to customize AS/400 to satisfy organizational requirements. Because these options affect system security, only the designated security officer is allowed to modify these values. The authorities of a security officer are assigned in the user profile, as described previously in Exhibit 3.

Security-Related System Options

Security-related system options support the following four types of controls:

- **Sign-On Controls** These controls determine how the system handles attempts to sign on, including limits on reentry of invalid passwords, system action when this limit is exceeded, and restrictions on the devices that the security officer and other privileged users can use to sign on. To discourage users from leaving a workstation unattended without first signing off, controls can be set to limit simultaneous sessions from multiple devices.

- **Password Controls** These controls determine how often passwords must be changed and permit establishing rules for password content to prevent the use of trivial passwords.
· **Active Job Controls** These are the values used when explicit security values have not been specified. For example, a default value can be used to specify the time period for deactivating inactive terminals.

· **Audit Controls** These controls determine the types of events recorded in the system audit log (referred to as an audit journal). Users who have been designated as auditors can change the audit related system values.

A complete listing of system options for sign-on, password, active job, and audit controls is provided in Appendix B, along with recommendations for selecting the values for these options.

The next section describes how to set system values using the QSECURITY system option for sign-on control as an example. The format for entering commands described in this section is similar to that for the other system options listed in Appendix B.

### Setting the System Security Level

The QSECURITY system option is used to establish the level of security required by the organization. AS/400 supports four increasing levels of protection. Each higher level of protection requires additional time and resources to manage the security of objects. Therefore, the IS and security managers must attempt to establish a level of control that is secure without sacrificing system efficiency. The following values, listed in ascending order of security protection, can be selected for this option:

- **No Security (System Value=10).** No password is required to sign on, and a user profile is created automatically if none exists. Users can access and delete all objects; accidental deletion of data is not prevented. This level of security is not recommended: it should be considered only for installations in which there are a small number of trusted users and in which physical security measures are adequate.

- **Sign-On Security (System Value=20).** The user must enter a valid password to sign on, and a user profile must be created. Object-level security is not enforced, although applications can restrict users to a limited selection of menu options. Security management requires only enrolling and removing system users and maintaining menus. This minimal security level is recommended only for installations in which access to data is controlled by menus and in which no access is granted to query and other data utilities.

- **Resource Security (System Value=30).** This security level enforces user access to objects. Menu security can be used, access to objects can be restricted by library, access to objects can be authorized using user profiles, or public access can be granted. To reduce the time required for security management, objects can be grouped in a library and access to the library can be defined.

- **Resource and Integrity Security (System Value=40).** New AS/400 systems are shipped with this default setting. In addition to the features of level-30 security, this security level prevents the attempted circumvention of security by programs written in such languages as C and Pascal that employ the full machine interface pointer capability. There are no security management requirements beyond those for level-30 security. This security level is recommended for all installations; however, applications that use internal system interfaces or that access objects directly will not operate at level-40 security.
Enhanced Integrity for C2 (System Value=50). Systems that require the highest level of security can select level 50. This level is designed to meet the requirements defined by the US Department of Defense for C2 security. It provides enhanced integrity protection in addition to what is provided by security level 40. Security level 50 provides enhanced integrity protection for installations with strict security requirements. Level 50 is not recommended for most installations because the additional checking adds a 5% to 15% in CPU cycles.

CL commands are used to establish and maintain system values. To display the value of a system option, the DSPSYSVAL command is used. For example, to display the current value of the QSECURITY option, the following command would be entered:

DSPSYSVAL QSECURITY

The following CHGSYSVAL command changes the value of QSECURITY:

CHGSYSVAL QSECURITY '40'

In addition to these two CL commands, the WRKSYSVAL command can be used to both display and change system values. The WRKSYSVAL command can be used to obtain a full description of system options; the user is prompted for any further action to modify the values of these options.

After a change has been made to a system option, the new value takes effect only after the system has been restarted (i.e., after an initial program load has been performed). This delay is desired, given the significant impact that a change in these values can have on system security.

Defining Security-Related Network Attributes

Network attributes define the system characteristics of the network, including the system name and network security attributes. The attributes JOBACN, DDMACC, and PCSACC have a particular impact on network security. JOBACN specifies how batch jobs from other systems are handled—for example, jobs can be automatically submitted to the job queue for execution or rejected. DDMACC names an exit program that controls the processing of requests to access system files, and PCSACC names an exit program that controls requests from personal computers. Appendix X shows the values for the security network attributes. The work net attribute PCSACC option *REGFAC indicates the system uses a registration facility to determine which exit program (if any) to run. The registration facility can be used to improve system performance by only calling the exits that are required. If an exit program is named in the network attribute the exit program is called for every PC request rather specific request types.

CL commands are used to establish and maintain network attributes. DSPNETA can be entered to display all network attributes; CHGNETA is used to change these attributes. For example, the following CHGNETA command is used to assign the exit program DDMAUDIT in the SECURITY library to the network attribute DDMACC:

CHGNETA DDMACC(SECURITY/DDMAUDIT)

Only the security officer is permitted to change the security-related network attributes.

Registration Facility

The registration facility is a service that provides storage and retrieval operations for OS/400 and non-OS/400 exit points and exit programs. An exit point is a specific point in a system function or program where control may be passed to one or more specified exit
programs. An exit program is a program to which control is passed from an exit point. The registration facility includes exits for the following security-related functions:

- Client Access (PC Attachment)-Exit programs for file transfer, remote commands, printer, direct access to objects through the integrated file system.

- User Profiles-Exit programs for user profile creation, change, deletion and restore.

- File Transfer (FTP)-These exit points are used to set up additional security and validation controls for the file transfer protocol (FTP). The FTP client and server request validation exits are for controlling the use of FTP subcommands. The server logon exit point authenticates the user who is trying to logon to the FTP server. Also, you can use the two server exit points to establish an anonymous FTP server site.

**Authorization Lists**

An authorization list is an object that lists users, their authorities, and the objects to which the members of the list are authorized. When users are added to the list, they become immediately authorized to access all of the objects on the list. Exhibit 6 provides an example of an authorization list. In this example, the users Evans and Grant are listed, as is the GRPSALES group. If Evans and Grant both belong to the GRPSALES group profile, what authorities would they be granted based on the information provided in Exhibit 6? Because access granted to individual user profiles takes precedence over that of group profiles, Evans retains more authority than other members of GRPSALES to the objects secured by the authorization list (i.e., *ALL authority exceeds *CHANGE authority), whereas Grant has less authority (*USE authority is more restricted than *CHANGE authority).

**Sample Authorization List**

The CL commands used to create and modify the authorization list are provided in Appendix A. The following CRTAUTL command can be used to create the authorization list shown in Exhibit 6:

```
CRTAUTL AUTL(SALES1) AUT(*EXCLUDE) TEXT('Sales Department')
```

These ADDAULTLE commands can be entered to add the user Grant and the GRPSALES group to the authorization list with the authorities shown in the exhibit:

```
ADDAULTLE AUTL(SALES1) USER(GRANT) AUT(*USE)
ADDAULTLE AUTL(SALES1) USER(GRPSALES) AUT(*CHANGE)
```

There are several advantages to using an authorization list to secure objects rather than using individual user profiles. First, the same authorization list can be used for multiple objects, which reduces the number of authorities that must be stored. The fewer the number of authorities, the faster it is to back up the system.

Second, the name of the authorization list for an object is automatically recorded when the object is saved. If the object is restored to the same system, the object is reconnected to the authorization list. However, if the object is restored to another system, the authorization list is not automatically attached, because of the possibility that users may be different on the other system. (In the latter event, the security officer must restore the object using the ALLOBJDIF(*YES) parameter of the RSTLIB or RSTOBJ command to allow the authorization list to be attached.)

Third, the security of a file cannot be changed when a file is open by any job. Often, an installation has some critical files open for extended periods; the security officer at such an
installation needs to find a time when no one is on the system. This is inconvenient and often requires changes at late hours. If an authorization list is used to secure the file, the users on the authorization list can be changed while a file is open. This practice allows the security officer to make security changes in a timely manner.

Last, objects attached to a library are secured by an authorization list, as specified by the library attribute. This feature eliminates the requirement of having the security administrator authorize all newly created objects.

**Adoption of Authority by Programs**

A program can be assigned an attribute that permits users of the program to gain the authority of the program owner while running the program. This feature eliminates the need to specifically authorize users to the objects references by the program. The internal program logic may, of course, enforce limits on the operations the user can perform. For example, the program may restrict the processing of transactions that exceed $1,000.

Program adoption of authority can simplify security management. However, programs that adopt the owner's authority must be carefully designed to eliminate potential security exposures. Access to such relatively unrestricted functions as command entry and query facilities should be avoided. (For example, if the adopting program were to present the user with a command line, the user would be able to access any object to which the program owner was authorized.) Adopting programs should also provide sufficient controls over the job's library search list to prevent the addition of a library containing a program or command designed to circumvent security. The design of programs called by the adopting program also must be examined, because the adopted authority is shared by all called programs.

The following CRTCLPGM command can be used to create the CL program SAMPLE that adopts the owner's authority:

```
CRTCLPGM PGM(SAMPLE) USRPRF(*OWNER)
```

The USRPRF(*OWNER) parameter specifies that the program is to adopt the owner's authority. The DSPPGMAD command is used to list all programs that adopt the authority of the user specified with the command. The output from this command can be stored in a data base file for audit purposes.

The default for programs is to use adopted authority from previous programs in the stack. Programs can be changed with the Change Program (CHGPGM) command or Change Service Program (CHGSRVPGM) command to set the USEADPAUT parameter to *NO, which will prevent the use of previously adopted authority. A system value Use Adopted Authority (QUEADPAUT) can name an authorization list. When the system value names an authorization list, only the users have *USE authority and can create programs that use previously adopted authority.

**Authority Search Order**

As demonstrated by the preceding discussion, AS/400 supports a variety of mechanisms for assigning authority. During processing, the system follows a prescribed order of search among these authorization mechanisms to determine if an operation is permitted. The search stops at the first authority found. The search path and logic are shown in Exhibit 7.

As shown in the exhibit, several factors define the precedence of authorization:

- Access specified in the individual user profile overrides that in the group profile.
- Access granted to a profile overrides that in the authorization list.
Public access is used if no other authorizations are found.

Authority Search Order

Establishing a Security Strategy

As discussed, AS/400 provides various mechanisms for securing data and applications. Access can be restricted through use of menus, library security, and object security. Each of these methods of protection provides different levels of security; in most installations, no single method is sufficient to cover all security requirements. Therefore, in defining a security strategy, a combined approach that uses all of these methods is recommended. The following sections briefly review the set-up requirements for using the menu, library, and object security methods and then discuss how these methods can be used in combination to provide optimum protection for all system users.

Menu Security

Menu security offers a very simple security strategy. Upon signing on to the system, users are restricted to selecting processing options from a menu. The menu can be designed to provide only those options required for performance of job duties—for example, predefined queries or file transfer requests. Users do not need to be authorized to access individual objects.

Menu security can be used in systems in which the QSECURITY system option is set to a value of 20 or higher. The user profile options INLPGM and INLMENU are used to define the screen the user first sees when signing on. The user profile option LMTCPB should be set to *YES to prevent the user from changing the initial menu and program on the sign-on screen.

The protection offered by menu security is limited. Menu security does not prevent a PC user from accessing data by file transfer, remote commands or the integrated file system. Library or object security should be used in addition to menu security to properly protect information.

Library Security

Library security protects objects by grouping them into a library and restricting access to the library. Only users authorized to the library can access objects in the library. The objects themselves are assigned *PUBLIC access authority. To implement library security, the QSECURITY system option must be set to a value of either 30, 40, or 50.

Object Security

Object security refers to the granting of access to individual objects. To gain access to the object, users must be authorized to both the object and the library in which it resides. Although object security provides the highest degree of separation of users according to access authority, its management can be excessively time-consuming unless group profiles and authorization lists are used. This type of security is recommended only for such objects as payroll files and other sensitive information that requires the highest level of protection.

An Integrated Approach to Security

Because access requirements differ among different types of system users, the organization’s security needs are generally best served by using a combination of security
approaches. The following recommendations are based on an analysis of the security requirements of different groups of system users.

General system users should be restricted to using menus to perform such predefined sets of tasks as file transfers. More skilled users, however, often require the use of system utilities to perform such functions as data base queries of the downloading and uploading of files. Allowing these users direct access to system functions can help eliminate a backlog of processing requests to the centralized IS department. However, menu security measures are inadequate because query and other system facilities permit unrestricted access to all objects. To control access to objects, library security and, where necessary, object security must be employed. Skilled users should be given *USE authority to files in order to restrict them to printing the results of the query rather than storing the results in a file. The user profile option LMTCPB should be set to *YES to prevent the user from entering system commands.

Programmers can be restricted from changing objects in the production environment through the use of library security techniques. Programmers should be given individual libraries for application development. The development process should be controlled using change management facilities. Release of completed applications to the production environment should be similarly controlled, and both the source and the executable programs should be stored.

It should be noted that because system commands are objects, it is possible to restrict access to these commands to selected users. For example, to prevent users from using the TFRSECJOB system command to transfer processing to a secondary job, the following CL command can be executed to revoke *PUBLIC access:

```
GRTOBJAUT OBJ(TFRSECJOB) OBJTYPE(*CMD) USER(*PUBLIC) + AUT(*EXCLUDE)
```

The user Grant can be given limited access to the TFRSECJOB command, as follows:

```
GRTOBJAUT OBJ(TFRSECJOB) OBJTYPE(*CMD) USER(GRANT) + AUT(*USE)
```

**AS/400 Audit Support**

The security manager and EDP auditor are usually most interested in answers to these two questions:

- What are the current settings of the AS/400 security parameters?
- What security-related events have occurred over a period of time?

The most recent releases of the OS/400 operating system have offered significant improvements in the mechanisms provided to answer these questions.

**Audit Reports**

The CL display commands contained in Appendix A can be used to list security-related information. (The display commands are denoted by the DSP prefix.) When used with the OUTFILE options, these commands retrieve information to a data base file. Data base query utilities can then be used to produce reports containing such information as:

- All objects authorized to a user profile.
- All user profiles authorized to an object.
User profiles that have a high level of access (e.g., *ALLOBJ authority).

- Objects secured by an authorization list.

- User profiles that belong to a group.

To reduce the volume of data that must be reviewed at any one time, the results from previous periods should be saved. These results can then be compared to the results of current analysis in order to identify differences.

**Audit Journals**

A journal is an object that is used to record data in such a way that it cannot be modified. An audit journal is used to store such security-related events as access failures as well as most types of successful access operations, including the creation and deletion of objects and security-related changes to object authorizations. The security officer uses the QAUDLVL system option to specify the types of events to be recorded. The values permitted for this option are shown under the Event Audit Options heading in Appendix D.

Data contained in the audit journal can be retrieved to a data base file. Data base query utilities can then be used to summarize this information for analysis and reporting.

The audit features have been extended to meet the C2 audit requirements. All security-related changes can be audited. Audit can be selected for all or individual users, and the actions of users or reference to objects can be audited. However, excessive auditing can affect system performance.

**Recommended Course of Action**

The first step in securing the AS/400 system is to identify critical data files and applications and establish the appropriate level of access control over them. At most AS/400 installations, system security should be set at level 30 to 40 to ensure adequate protection of resources. To optimize system performance, *PUBLIC authority should be used when possible; otherwise, objects with similar security requirements should be grouped in a library to which access is restricted. To simplify the enrollment and removal of users, individual user profiles should reference group profiles whenever possible.

Certain controls common to most system installations should also be considered. Password expiration and validation system values should be implemented to enforce the periodic changing of passwords and to prevent the use of trivial passwords. A timeout period should be set for inactive terminals and vendor-supplied passwords should be changed to prevent easy access to the system by hackers. In particular, passwords to the following vendor-supplied profiles should be changed: QSECOFR, QSRV, QSRVBAS, QPGMR, QSYSOPR, and QUER.

For most organizations, an integrated strategy using menu, library, and object control mechanisms can provide an efficient and cost-effective solution to their security requirements. This strategy should be augmented by a security awareness program that stresses the role of system users in protecting information assets. Any implementation of new or modified security controls should be explained in a positive manner so that users accept the need for such changes and appreciate the importance of secure access to company information.

**Author Biographies**

Wayne O. Evans

Wayne O. Evans is an independent security consultant specializing in the AS/400. Previously, he worked for 27 years at IBM Corp., where he designed the security controls
for the System 38 and AS/400 series computer systems. Evans is a frequent speaker on information security topics.
Authorization List

SALES 1

Owner
EVANS

Users
GRANT
GRPSALES
*PUBLIC

Authorization
*ALL
*AUTLMGT

Authorization
*USE
*CHANGE
*EXCLUDE

List of Users and Authority

List of Objects

SALES *FILE

SALES *LIB

CUSTMASTER * FILE

DAILYSUM * PGM