Access control systems: From host-centric to network-centric computing

by M. Benantar
R. Guski
K. M. Troidle

In this paper, we review the important theoretical models of computer access control systems. Using the IBM Multiple Virtual Storage (MVS) operating system as the example, we show how the principles expressed in these models can be implemented. The roots of user authentication and access control on MVS are examined, tracing the convergence of the requirement for controls with the development of appropriate software. The paper also highlights security and auditing features unique to the host-centric computing model and discusses how these existing technologies might be applied to the security problems presented by the newer networking computing models. Finally, we look to the future and suggest the environment that will result when the host-centric computing model for security and access control is interoperated with the networking computing models.

The *host-centric* computing model emerged in the late 1970s and involves a mainframe computer where most—if not all—useful processing occurs, and to which most data are connected. End users access the computing resources of the mainframe via a set of terminals. Early versions of such terminals were capable of only text-based interaction with the human user of the system. Today, these are giving way to the widespread use of personal computers as terminals, thus supporting the use of graphical human interfaces with the host-centric model, but with most of the processing still occurring within the host. Note that the host-centric model may support several different applications (or services) simultaneously, and those applications may optionally interface with each other according to the needs of the end users.

In contrast to the host-centric model is the *distributed* computing model, in which some set of individual computers are networked together, generally in a peer-to-peer arrangement. In this model, processing power is distributed among the various individual machines in the networked set of machines so that any one machine may call upon any other machine in the network, for some particular computational service or element of data offered by that machine. Client/server computing is a variation of the distributed computing model in that certain machines in the distributed networked set are designated as *servers*—of some specialized computing service or shared set of data—with the rest of the machines in the set being potential *clients*. Human interface processing is done at the client machine, usually called a *workstation*, close to the user. Today, distributed networks often include several server machines that provide various services to numerous individual computing system users at their individual client machines.

The so-called *network-centric* model is the newest computing model. From the perspective of processing power, the network-centric model is seen to embed the processing power of large-scale multiuser server systems within the open public network. Clients do not have to know who or where the server is that will process their request for information or processing power. They simply ask the network for the service (or perhaps for a subscription to it) from a public directory. In contrast to the distributed client/server model in which some level of cooperative processing power must exist in the user's workstation, the level of processing capability that the client system has in the network-centric model is optional depending on the application, and can range from the processing power of a high-end workstation all the way down to the minimal processing power of a common telephone. The network-centric model represents a significant opportunity for vendors of large-scale computing systems, since a powerful multiuser server differs little from other examples of the traditional host-centric computing model.

Whether it be host-centric, distributed, or network-centric, the computing system that the end user sees is one that is being shared by more than one user, and whenever any computing system is subject to use by more than one user, issues of information privacy and control of computing system resources and information emerge. The need to address these issues effectively in the newer computing models is a serious challenge and potential inhibitor to the industry, and therefore an opportunity for the current operating system vendors. These issues and their resolution through the implementation of computer security and access control systems is the topic of this paper.

At first it may seem that addressing these issues with the development of an internet World Wide Web server computing system application, for example, will be different from previous experience, requiring new inventions to apply to the problem. A closer look suggests that designers and developers of modern distributed and network-centric computing systems should look at the information privacy and security solutions that have been applied in the past to the host-centric computing model that is perceived, at least, to have better security and auditing capabilities than the newer computing models. Indeed, the predicted rapid migration of business data processing applications off of the mainframe system and onto distributed computing systems has not occurred, with security concerns often cited as the main reason as a cause for this delay.

With this objective in mind, we proceed in the next section with a review of the basic theoretical model of computer access control principles. Then, using IBM's Multiple Virtual Storage (MVS) operating system and Resource Access Control Facility (RACF*) as examples, we will show how these principles have been applied to the host-centric computing model. In subsequent sections we will discuss the basics of administration of the access control system, and introduce the concepts of identification and authentication, resource access authorization checking, and auditing. In addition, we will call attention to some specific access control functions that are possible with the host-centric computing model but either do not exist or are difficult to implement on the other computing models.

Whenever any system is being shared, issues of privacy and control of resources emerge.

The Lampson reference monitor model

Modern access control mechanisms are based on the *reference monitor* concept introduced by Lampson[1] (see Figure 1). A reference monitor is the component of the computing system that mediates every access of a subject to a resource in accordance with a security policy implemented in the form of rules and attributes associated with a registry of subjects (or users) and resources. A security administrator defines a subject to the computing system, i.e., making available to the user an *account* that is a representation of that user to the computing system. By possessing an account, which can be viewed as a logical *identity* within the system, a user acquires the ability to access the system and interact with the resources that are within its scope.

Similarly, a resource entry in the registry represents some system resource such as a file. A security attribute of a resource could be an identity of its owner, or of some user that is granted some type of access to the resource. In addition to mediation (as previously mentioned), this concept supports isolation of the security services from untrusted processes, thus avoiding tampering and maintaining integrity. The reference monitor also makes a convenient point for generating audit records that document users' actions (access to resources) in the system, as we will discuss later in more detail. Figure 1 illustrates the Lampson reference monitor concept. Note that the user and resource registries both reside within the boundary of the reference monitor and are therefore tamper-proof.

Shortly, we will describe how the Lampson reference monitor has been implemented on the MVS operating system. But before that, we need to discuss computing system integrity.

System integrity: A prerequisite of access control. The evolution of the MVS operating system can be traced back to the earliest days of the System/360* computer architecture. Interactive computing was not yet in general use. Large numbers of users did not share a common computing resource or data files. Security requirements focused on the physical security requirements of the computer system installation. There was no support for an access control reference monitor.

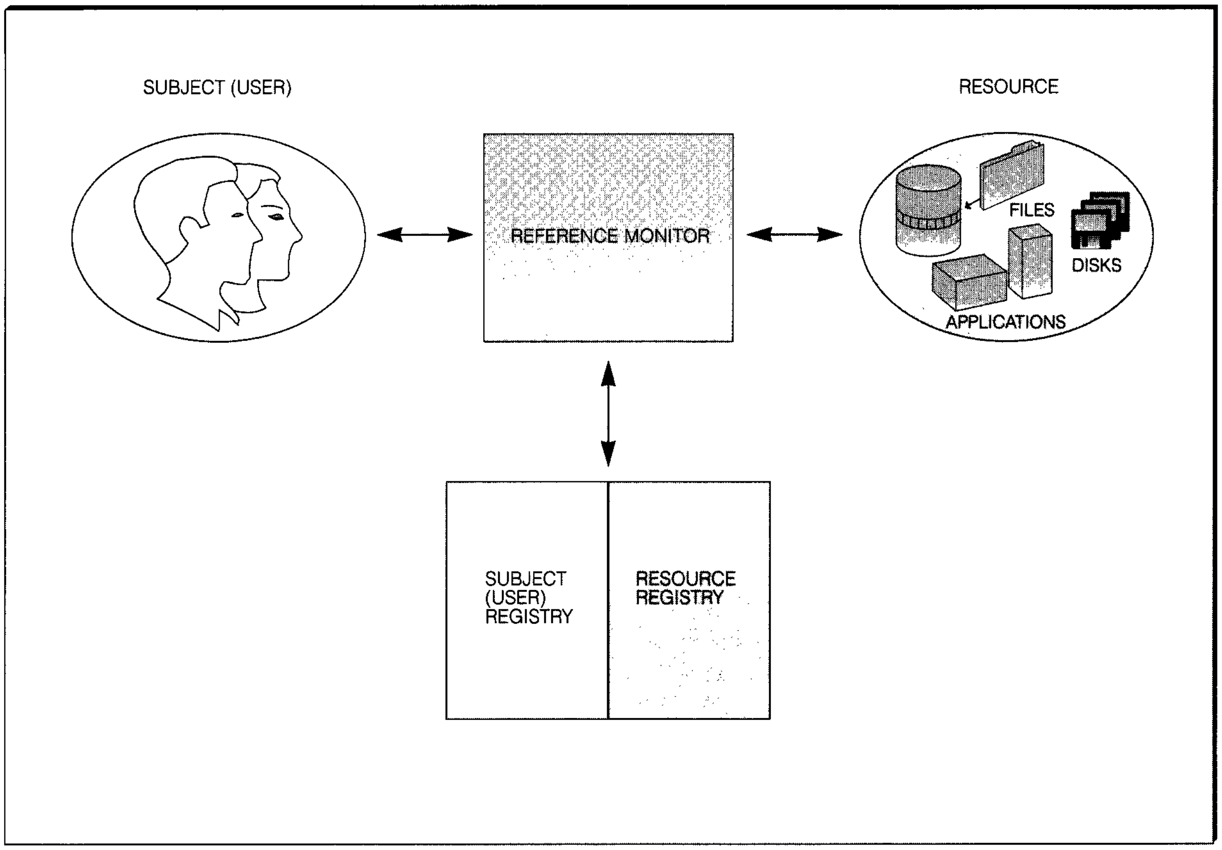
System integrity rather than security. Rather than security and access control, the *integrity* of information within the early general-purpose computing systems, such as IBM's System/360, was the focus of attention. For example, the execution of multiple processes concurrently within a common memory meant that a given process might, in error, overwrite memory assigned to another process. This problem was addressed with storage protection keys; a particular process and the storage assigned to it are assigned a unique storage key that must match if the process is allowed to access the storage. Any attempt by a process to store data outside of its own area of memory is recognized in the hardware by mismatched storage protection keys.

Similarly, multitasking requires a control program that can orchestrate the execution of other tasks. In System/360 through System/390* architectures, the control program is isolated from user programs by means of a two-state instruction execution environment. These two states are called *supervisor state* and *problem program state*. A special set of machine instructions, including input/output (I/O) commands to the I/O channels and memory and address space management instructions, are operable only when the computer is in supervisor state. The control program typically executes in supervisor state while end-user programs always (providing the system is configured properly) execute in the problem program state. When an end-user program requires the services of the control program, for example to do I/O, it makes a request to the control program. The control program—before executing the requested function, and now executing in supervisor state and beyond tampering by the problem program—examines the request to make sure that it will not exceed the logical boundaries of the problem program. For example, a logical boundary might be the ability to read or write data to a storage area that has been allocated by the data management component of the control program, to some other problem program or to the control program itself.

MVS system integrity statement. Two-state hardware and storage protection keys, plus other functions such as MVS address space isolation, keep processes separate from one another and are the technical foundation of MVS system integrity, making up the *MVS Integrity Statement*. This statement is a commitment by IBM to provide corrections to technical problems that are found to compromise the integrity of the operating system. The integrity features of System/390 and MVS are important, of course, to all aspects of business computing, but to an access control and security software package, they are an absolute prerequisite. The security software depends on the integrity features to ensure that the implementation of a security policy cannot be compromised by some action of a problem program that happens to also be executing in the same machine (perhaps under the control of an unauthorized user). System integrity is the foundation for access control software.

Since the integrity features of System/390 are controllable by operating system software, the integrity of the MVS control program can be extended to cover a new component, which, as we will see, can include an external security software package. When such a component is installed, it becomes just another part of the control program. The implementation of the Lampson reference monitor concept for MVS has occurred through extensions that have been made to the MVS operating system in the form of access control software packages such as IBM's RACF, and Computer Associates' CA-ACF2** and CA-TopSecret**.

Figure 1 The reference monitor model



Mapping the Lampson model onto MVS. As the machine capacity evolved to support the work of more and more processes concurrently, so did the need for access control. IBM's RACF was introduced into MVS in 1976 as an external add-on product.

Mapping the Lampson registry with RACF. In accordance with the Lampson model, RACF consists of a registry of users (or subjects) and resources. The information contained in the registry describing each user is sufficient for RACF to determine the user *identity* within the overall population of users. The information describing resources is such that the RACF can make an algorithmic decision as to whether or not some particular individual user has authority to access some particular resource, basically a simple matter of determining *who* has access to *what.*

Access control decision making. Information maintained in the RACF registry is used by RACF during various processes that provide access control decisions and other services for requesting resource managers. In general terms, these functions include:

- Accepting user processes into the system and keeping track of them while they are within the system
- Processing resource authorization checking requests for user to resources
- Detecting and responding to auditable events

Each of these will be covered later in more detail.

The role of the resource managers. In addition to the registry of users and resources and support for the decision-making process, the reference monitor must include some method for the access control software to know when a *security-relevant event* such as a user accessing a file, is taking place. In the MVS case, this is the responsibility of the code that is in control of the actual physical resource. For example, the data management component of the operating system is the component that is responsible for managing the access to individual data files stored within the computing system, and is referred to as the *resource manager* for data files. Such operating system components and their equivalents in applications software, are known as *resource managers*. Examples of MVS resource managers include the data management system, known as Data Facility Product (DFP, which is the equivalent of a file system in UNIX** terms), Information Management System* (IMS*) for transaction management, and MVS Contents Supervision (which causes programs to be loaded for execution). In the Lampson model implementation on MVS, resource managers are viewed as components of the reference monitor. The function of the resource managers in this context is to invoke the services of the access control software whenever a security-relevant event occurs, such as a user logging on to the system or a user accessing a data file.

There is a need for a centralized access control manager that can be used by different servers.

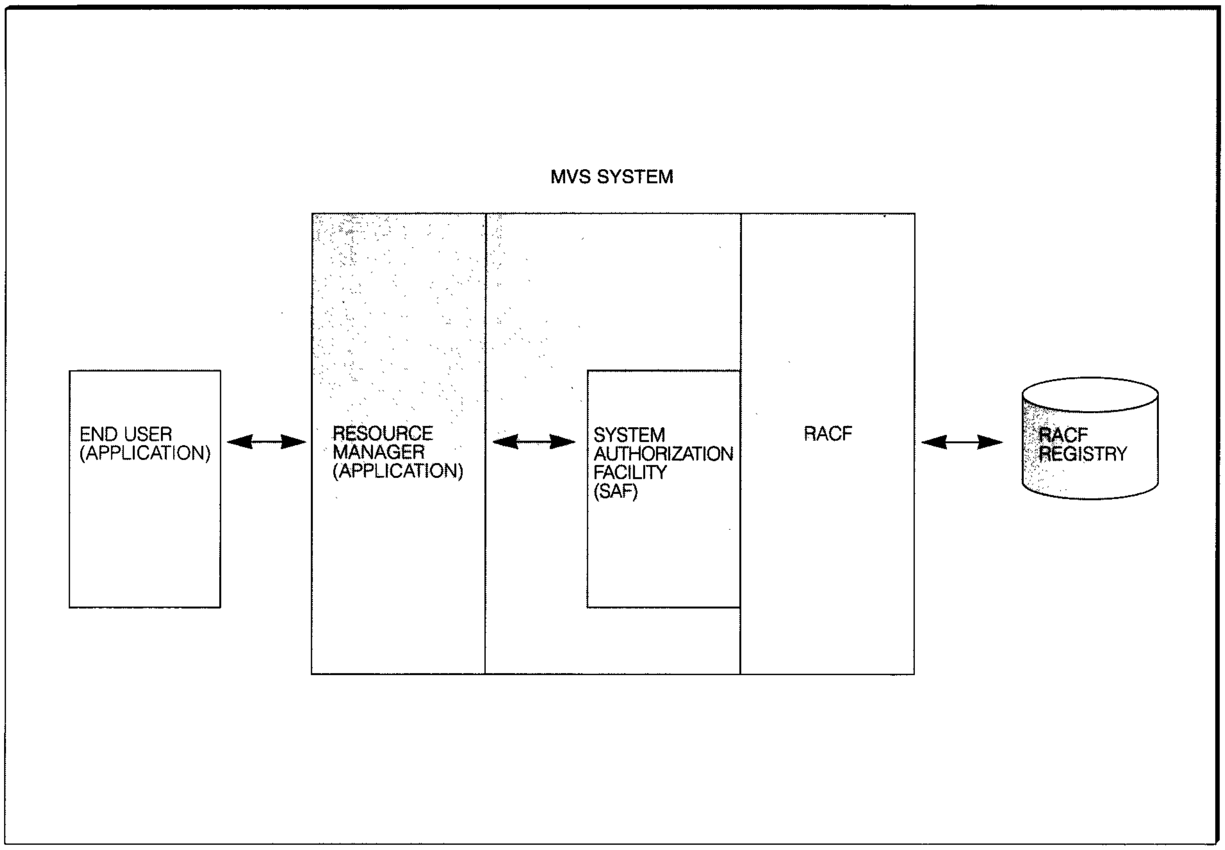
MVS resource managers invoke security via SAF. Early use of RACF services by MVS resource managers was through an MVS-supplied macro language designed to interface exclusively with RACF. Later this set of macros evolved into the MVS System Authorization Facility (SAF), which was generalized into an interface that could support access control packages other than RACF.

Therefore, with MVS, it is the combination of system integrity, the SAF interface, and the external-to-MVS access control software package, that constitutes the Lampson reference monitor model implemented on MVS. Although others have written of the advantages of designing all components of the Lampson reference monitor so that they reside within a single area of the operating system (usually referred to as the *security kernel*[2-4]), in practice it does not make much difference that the components are in different parts of the operating system or within extensions of it. This is true providing that the integrity features of the operating system extend out to include such previously mentioned components and the overall size of the reference monitor does not get too big or too scattered for it to be analyzed and trusted. Thus, as illustrated in Figure 2, access control software is given control (via the SAF) by the operating system component managing the resource.

Resource access in MVS is monitored through a single point, i.e., the SAF component. Access decisions are made by a common process within the access control software package, instead of being decided by each resource manager. Resource manager application developers write code using the common MVS SAF interfaces without having to worry about which, if any, access control package is present on the system. In addition, the SAF interface on MVS offers flexibility in that enterprises can have their choice of access control software. Although we have used RACF as the access control system example in this paper, MVS installations have the option to use CA-ACF2 or CA-TopSecret as their choice of access control software package. All access control software is dependent on the SAF interface.

At this juncture, it is worth noting that the paradigm in the current Open Software Foundation Distributed Computing Environment** (DCE) model of computing is that an application server provides for its own access control services for the resources under its control. In addition to the issue of access control software redundancy versus reuse, multiple occurrences of access control code make the environment more susceptible to security breaches. This also undermines a desirable goal in system security design, and that is to minimize the size of the security component of a system. By having the application servers provide for their own access control mechanisms, the system security component grows in size and becomes scattered. Consequently, server application developers are realizing the need for a centralized access control manager for a host machine that can be used by different application servers. Note that in such a scenario, RACF services can be readily made available to DCE application servers once the DCE services are deployed on an MVS system.

Figure 2 Interaction flow for resource access control in MVS


MVS SYSTEM
END USER (APPLICATION)
RESOURCE MANAGER (APPLICATION)
SYSTEM AUTHORIZATION FACILITY (SAF)
RACF
RACF REGISTRY

Administration of access control systems

In the previous section, we showed how the abstract components of the Lampson reference monitor computing system access control model map into functions provided by RACF. In the next several sections, we will cover various aspects of this implementation. This section focuses on the administration of the access control system (including the registry), who is authorized to update the security information, and the tools (human interfaces) they use to do so.

Implementation of security policy. The purpose of any access control system is to assist in and monitor the implementation of the security and resource access control policy on computing systems. Implementation of the security policy is the responsibility of the access control administrator, while the auditor is responsible for monitoring the implementation to ensure that the administrator is doing it properly.

Separation of duties of administrator and auditor. Although sometimes it becomes necessary for these two distinct functions to be performed by the same individual, host-centric access control systems such as RACF allow for the separation of duties of the administrator and auditor. A user who has been designated as an auditor cannot perform administrative activities such as defining users, groups, and resources. The auditor can, however, control the selection of *security-relevant events* for which auditing records are to be logged, and these events can include the actions of the administrator. To complete this separation of duties, the administrator cannot alter the selection of security-relevant events that has been made by the auditor. Thus the auditor can look over the shoulder of the administrator without the administrator being able to control what the auditor chooses to look at. This technical capability to separate the duties of the administrator and auditor makes it difficult for any single individual to compromise the implementation of the security policy.

The data files that are created as a regular part of end-user activities and normally accessed only by that user, are said to be *owned* by the user. A user who is designated as the owner of a resource, will usually have, by default, certain administrative privileges over that resource, similar to the privileges of the administrator and auditor. For example, in the case of RACF, an owner of a resource can permit other users to access the resource or can set certain basic auditing criteria for the resource. Although infrequent, these activities involve the use of access control system human interfaces by end users.

The user and resource registry. Consistent with the reference monitor model of enforcing system security controls, the access control software maintains a registry of users and resources as well as the security policy governing the use of those resources.[5] Access control software available on the market today handles these concepts in different ways. The rest of our discussion focuses on the MVS operating system and IBM's RACF. In this scheme, individual elements of the RACF database are called *profiles*. Profiles can be created to represent a *user*, a *group* of users, or a *resource*.

User profiles. RACF user profiles can consist of multiple parts called *segments*. There is always a *base* segment. Optionally, other special-purpose informational segments may exist that are logically related to the base segment. We discuss the base segment first.

Each user profile contains, at a minimum, a base RACF segment that includes information to uniquely identify the user (the user ID), the name of the user, and the name of each group to which the user belongs, as well as the corresponding authority the user has in that group. RACF requires that there must be at least one default group for each user. Also stored in this profile is an encrypted derivative of the user's password (which will be discussed later) and possibly one or more RACF attributes indicating special authority required for system administration or auditing roles (such as the SPECIAL, which in the RACF case denotes the administrator role, or the AUDITOR attributes).

In addition to the basic user information described, RACF user profiles can contain information segments for various MVS subsystems such as the MVS Time Sharing Option (TSO) segment that can be used by RACF to set up the security and processing environment for a user while accessing the TSO system. Other examples of subsystem specific segments in RACF user profiles are the Customer Information Control System* (CICS*) segment and the NetView* segment. A more recent example is the support for the MVS Open Edition* segment (OMVS). This defines attributes for users that allow them to execute applications or define and access resources in the POSIX** environment, which is defined by a set of standards for a portable implementation of UNIX that can operate across multiple operating system platforms that support the POSIX standard. Among these attributes we find the user identification (UID), the user's working directory (HOME), and the attribute indicating the path name for the shell program (PROGRAM) to be started when the user logs onto MVS OpenEdition.

Groups of users. In addition to information records describing individual users of the computing system, the administrative registries of access control systems contain records describing *groups* of users. A group name is a way of collectively identifying a set of users with similar duties and access control characteristics; e.g., grouping by department or work group for purposes of ease of information sharing, as well as simplifying the administration of access control. In the case of RACF, groups are logically related to one another in a hierarchical inverted tree structure. Each group profile contains the name of the superior group from which the current group is descending in the tree structure.[6]

The relationship between groups conveyed by the tree structure is used by RACF to support the ability to decentralize administrative authority. In a highly centralized administrative implementation, users with administrative privilege have authority over all users and groups defined to the registry, and this can lead to various problems. The inverted group tree structure of RACF allows selected users to be assigned administrative privilege that is effective only at a particular group in the tree structure and, optionally, to other subgroups, but not to the rest of the groups and users defined in the registry. This property can be used by the global administration team to carve out islands of decentralized administrative scope as needed within the global RACF user population.

Control of the security processing environment. A modern host-centric computing system can provide a wide variety of services and functions to its user population. This results in a similar variety of resources that may need to be controlled according to management's security policy. This requires extensive flexibility on the part of the access control software. Flexibility invariably results in a security-processing environment consisting of a complex set of security-processing options and selections. Effective administration of such an environment requires an understanding of the implications of these options and an ability to manage the intelligent use of them. Host-centric access control systems, such as RACF, include various controls for use by the administrator and auditor to assist with this task. In addition, monitoring and reporting facilities are provided to help the administrator and auditor to view the overall security-processing environment and to know when the security-processing environment or the underlying system integrity components have been modified (and possibly compromised).

Human interfaces to access control administration. One of the most important aspects of any computing system application is the set of human interfaces provided by the application. The human activities of doing access control administration on a daily basis involve the defining of users, groups, and resources to the system. Auditing responsibilities will involve the establishment of auditing criteria and the generation of audit reports documenting the status of the implementation of the installation's security policy. In addition to the activities of the administrator and auditor, the access control system, for example, enables end users to permit other users to access resources owned by the end users themselves.

Ease of use for a mixed audience. In the case of an access control system, the success of the system as a tool will be measured largely based on its *ease of use* by the administrator, auditor, and end user. Appearing *user friendly* to this mixed set of users adds difficulty to the job of the human interface designer. The administrator who uses the system every day, and is therefore familiar with its functions and syntax, will usually prefer a terse set of interfaces that can be used quickly and efficiently. On the other hand, the end user who uses the system infrequently will need extensive menus and help facilities. Between these two extremes exist all possible levels of user competency that must also be well served by the human interface.

RACF commands and panels. Because host-centric access control systems evolved long before modern computer graphics technology, the human interfaces employed were, and in most cases still are, based on command line technology. This is the case with RACF, which supports administration with a set of TSO commands. This command set is augmented by a set of Interactive System Productivity Facility (ISPF) panels and menus. In general, the line commands are strictly *function* oriented and are often favored by the experienced RACF administrator. The ISPF panels conversely, are *task* oriented, often consolidating multiple low-level functions into a single task that is more easily understood by the infrequent or new user. Until recently, this form of human interface has been acceptable, at least, to the host-centric user market.

The need for a modern graphical user interface. The most visually striking, and arguably the most useful, innovation in modern computing is the graphical user interface (GUI). With this approach, computing resources are displayed to the user in the form of objects that can be manipulated as though they were physical objects, through the use of a pointing and selecting device commonly called a *mouse*. GUIs first became popular with the Macintosh** personal computers offered by Apple Computer, Inc. Soon the GUI was embraced by Microsoft Corporation in the form of its Windows** personal computer and workstation operating system products. Unfortunately, modern GUIs that support the host-centric environment have been slow to emerge. This is unfortunate because the GUI approach presents interesting opportunities to improve the ease-of-use characteristics of today's host-centric access control systems.

Security management across operating systems. So far, our discussion of access control system administration has been limited to the MVS host-centric computing model with RACF as the example of access control software. Administration of such an environment can be accomplished with the human interfaces supplied with RACF. However, today's enterprise computing resources consist of various environments such as Novell NetWare**, OS/2* LAN Server, and even other MVS systems that might be using another access control software package. RACF administrative interfaces cannot be used to affect these environments. The IBM Distributed Security Manager (DSM) is a new product family that addresses the need for a common cross product and cross operating view of security administration. It will also help the very large RACF installation that desires to present a more "business" view of its access control environment than is usually the case with a set of human interfaces that are very access control product specific.

Establishing a security context

In the previous section, we described how the access control system maintains information about users, groups, and resources that have been defined to the computing system. In this section, we will show how that information is used by the access control system when a user first accesses the computing system, in building a security information record that will serve as the hinge pin for all subsequent security and access control events pertaining to that user. But first, we will discuss how the password evolved into use as a key to access the computing system.

The password: key to an identity within the system. The first general requirements for security function in computing systems were the result of having the capability to share information files among multiple users of the system, either serially or at the same time. Users realized that they did not really want to share all of their files with all other users, but that they would really rather have the capability to share selected files with a selected group of users. This requirement was initially addressed with the simple file password in the form of a character string, known by both the selected user (or users) and the file management software, used as a *key* to the file. Any user attempting to access a file, protected with a password, is required to reveal the key—the password—to the file management software, which then validates that this is the correct key for the particular file.

Password protection of data files is simple, effective, and is still in use in isolated circumstances. But it was really apparent that passwords used in such a manner present problems of scale in password management for both the computing system and the end user. The system has to store and maintain the secrecy of a large number of passwords, and individual users must remember all of the passwords that they have for their numerous resources, as well as those shared with them by other users. This is further complicated by the need to change passwords, especially those that are shared. Finally, when a password-protected resource is shared (along with the password), the identity of the user accessing the resource is lost. With file passwords, the secret key to the resource is decoupled from the identity of the user, so individual accountability is not possible. These difficulties led to requirements for more sophisticated access control systems.

Aside from the scalability problems, passwords are relatively easy to implement as keys to computer resources. Instead, passwords have evolved into use as a key to an identity, expressed as the *user security context*, which conveys various authorities to the user within the computing system. To implement this fundamental principle of access control systems, individual users are assigned identities (user IDs) within the logical scope of the computing system control program or access control software. Users *sign on* to the system by asserting to be particular predefined identities (users) who have been assigned authority to access this particular computing system (and perhaps application) and then prove, in one manner or another, that they are who they claim to be. In traditional host-centric computing interactive applications, the *act of authentication* is accomplished by having users specify the correct password associated with their identity, as part of the sign-on sequence, so that it can be validated by the access control system. Thereafter, as long as the user has not *signed off* from the system, the user's authenticated identity remains logically connected to the user in the form of the user security context, and can be used for both resource access control and accountability (audit trail generation) purposes. This process, which is implemented in a different manner in the Kerberos** authentication protocols (which we will discuss later), is the foundation of all modern implementations of the Lampson model, and is known as the user identification and authentication process.

The user security context confines actions, allows accountability, and defines authority.

The user security context. In general terms, a user's security context carries the user's rank and group membership within the population of users defined to the access control system. The security context is used by the system to confine the user's actions in accordance with the privileges that have been defined to the identity (or userid) with which it is associated. It also represents the credentials that may be assigned to the user, and used to trace the user's activities for accountability purposes. The security context is also a logical anchor for any special authorities that the user may have, such as a particular system administrative or organizational role that the user may assume, as well as any groups to which the user belongs. Due to its sensitivity, the user security context is always protected from modification by users of the system. With rapidly evolving network-centric computing environments, the need arises for a network-wide security context. In the next section we touch on the emerging de facto standard, the Kerberos network authentication system.

Kerberos. In the emerging network-centric computing environment, powered by the wide availability of end-user desktop hardware, users are logically connected to the computing resources of the network instead of the single host environment. This has led to the introduction of alternative authentication methods that define the scope of an identity within the realm of a network environment. Simply put, the security context of an active entity such as an end user needed to be extended throughout the network. One novel method that is gaining wide acceptance is originally attributed to the Massachusetts Institute of Technology Project Athena**, known as the Kerberos set of protocols.[7,8] Kerberos features a third-party authentication server unique within the scope of an underlying network and is trusted by all active entities including end users, computing machines, and application servers participating in a distributed computing environment. The underlying trust is based on the fact that each such active entity reveals its secret key in the form of a password only to the network security server. In order to acquire a network security context, for instance, an end user at a client workstation identifies him or herself to the Kerberos authentication server. The latter uses its local registry to construct an *authentication ticket* and sends it back to the client workstation so that it can be used to gain access to other network services. The novelty in this protocol lies in the fact that the end user's password never flows over the network. Rather, the Kerberos system encrypts the ticket using the client's key, and upon receipt of the ticket it has to be decrypted by the correct password that the end user is prompted to enter. As a result, the user acquires a network security context in the form of an authenticated ticket that a client workstation passes to other network entities in the course of requesting remote services on behalf of the active user. With the physical security of the network security server, Kerberos protocols are generally very secure and are the basis of providing authentication and security in the evolving distributed computing environment (DCE).

Inherited and assigned identity with MVS. The user security context on MVS is called the Accessor Control Environment Element (ACEE). All system and user functions executing on MVS, including the MVS Master Scheduler (analogous to the *kernel* in UNIX or OS2*), have an ACEE and, therefore, an authenticated identity associated with them. ACEEs are protected from modification by end users. The SAF interface (the MVS System Authorization Facility) provides support for the creation and maintenance of a security context. Trusted operating system components and trusted server applications (but not end users) can invoke the appropriate service to create and later delete an ACEE associated with some identity, without requiring the presence of an authenticating password or key. This capability supports two important characteristics of MVS:

1. A new process that is initiated by an existing (executing) process can inherit the authenticated identity of the parent process.
2. A trusted process or server can initiate a new process, with the identity of some end user, without the logical or physical presence of the user or copy of the user's password.

An important point is that both characteristics 1 and 2 can be accomplished without the need to access a copy of the user's current password, as would be the case if the Kerberos authentication protocol was used instead. The risk and complexity associated with maintaining a current copy of the user's password or a hash derivative of it in order to support similar services are thus bypassed.

Because of these characteristics, the MVS Master Scheduler can drive the initiation (or booting up) of its various components, subsystems, and servers, many with unique identities, with little operator intervention, no password exposure, and without concern that identities will "time out" requiring reauthentication. Equally important: MVS server subsystems including IMS, CICS, and the Job Entry Subsystems (the MVS batch application servers), can initiate processes that have the identity of the end user on whose behalf the services are being executed, so that access control and audit functions will correctly associate access control decision making and auditing with the correct user. This capability is in contrast to the DCE environment in which an application server is required to explicitly engage in a third-party authentication process with the Kerberos security server in order for a security context (either that of the server itself or of the client) to be established. This process requires that the server retrieve and make use of its secret key *and* the secret key (password) of its clients. This introduces problems of secure password storage and synchronization when the server or clients (users) change their passwords.

Improved security for the password. The password, the most commonly used method of user authentication, has evolved some sophistication of its own, especially in the manner it is stored within the registry. In fact, with RACF, the password is not stored at all. Instead, using the password as an encryption key, the user ID is transformed by a one-way implementation of the Data Encryption Standard (DES) algorithm. The result is compared with the value stored in the profile that was computed likewise when the user last updated the password. When the two encrypted values match, the user has entered the correct password. There is no way to recover the original plain-text password from the stored computed value. This makes it impossible for a hacker, who may have somehow gotten access to the registry, to see the passwords of any or all users (which is an exposure in systems that store clear-text passwords). In addition, most host-centric access control systems include support for forcing users to change their password in accordance with installation criteria and software-enforced password rules that control the actual composition of user-selected passwords. The purpose of these is to avoid having users select passwords for themselves that are easy to derive by a potential hacker.

Although the security of password authentication techniques has improved, this simple authentication method suffers the disadvantage that the user enters a host-centric computing system password at the workstation, which subsequently flows over a communication line to the host—such as in a System Network Architecture (SNA) environment—in clear-text readable form. With users accessing host computing services from so-called *dumb* terminals, there was little that could be done to address this situation other than to coach or force users to change their passwords often since the passwords are so susceptible to compromise.

Alternatives to the password. As we have shown, in contrast to the Kerberos authentication protocol, which requires the actual password in order to decipher the initial encrypted ticket and complete the authentication process, in traditional host-centric access control systems such as RACF, the password is nothing more than an *authenticator* for users to prove that they are who they claim to be. The password is not otherwise *required* by the process of security context generation.

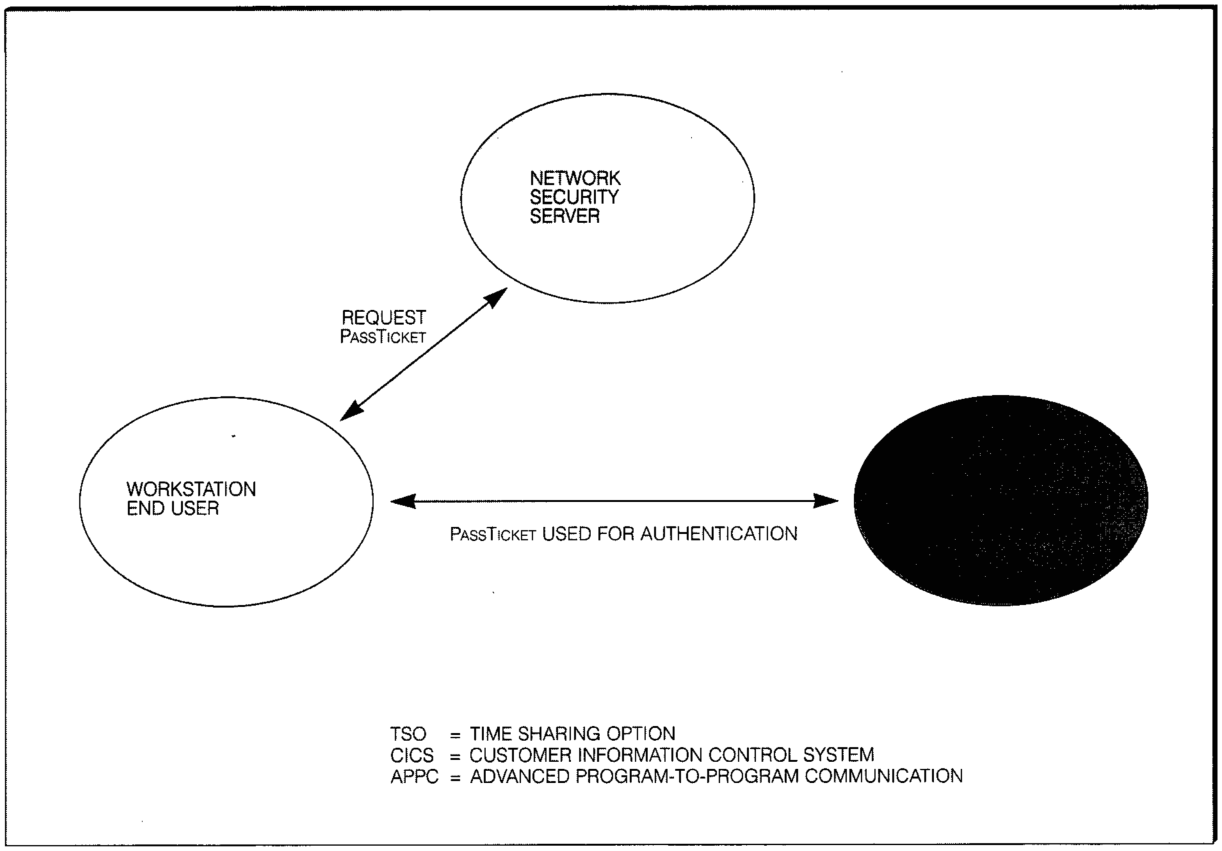
The distinction between the actual process of authentication and the creating of an authenticated identity (or security context) on MVS, has led to the emergence of authentication techniques that can be used as alternatives to the password. Several hardware authentication devices have been developed that have the user authenticate to the device, rather than to the access control software on MVS. This authentication can be as simple as entering a personal identification number (or PIN, which is a form of password),[9] or it may be complex, involving biological uniqueness of the individual, such as the dynamic of a signature or retinal scan. In any case, once the device validates the authentication, it communicates to the access control software that this user has already been authenticated and that the access control software can bypass that part of the identity creation process and proceed directly to the creation of a user security context.

It is even possible for this method of authentication to replace the traditional password entirely, thus bypassing the need for password content rules or functions to force users to change passwords periodically. The elimination of passwords would also address the problem of password synchronization for users who wish to use the same password to access different systems, perhaps on different platforms. Consider the time wasted by users, individually logging on to several systems, in order to update their password on each system when their password expires.

Authentication accomplished in this manner can be more effective and secure than the traditional use of a password, because the device can be something that the user *has* or something that the user *is* biologically, rather than just something that the user *knows* (the password). Merely having knowledge of something is generally thought to be more susceptible to being shared or inadvertently disclosed to others, with or without the consent of the possessor, as opposed to what someone physically has or is.

Another way to improve the way authentication occurs in the workstation and local area network (LAN) environment is the approach IBM has taken with the introduction of the RACF *PassTicket*, which is a more secure alternative to passwords while still using the existing password protocol. A PassTicket is a highly random, essentially *single-use-only*, password substi-

Figure 3 Use of RACF PassTicket in a client/server environment


NETWORK SECURITY SERVER
REQUEST PASSTICKET
WORKSTATION END USER
PASSTICKET USED FOR AUTHENTICATION
TSO = TIME SHARING OPTION
CICS = CUSTOMER INFORMATION CONTROL SYSTEM
APPC = ADVANCED PROGRAM-TO-PROGRAM COMMUNICATION

tute that can be generated for a user at a workstation and sent to a host system application, such as MVS TSO, for use during user authentication. A PassTicket must be generated in a secure manner, which can be by a third-party network security server, a callable service, or even by a *smartcard* such as the IBM Personal Security Card. The generation algorithm computes a PassTicket using the user ID, a key specific to the MVS host application, and a time stamp. These data are processed by the PassTicket generation algorithm using cryptographic transformations that result in a readable clear-text PassTicket similar in syntax to the RACF password. Once it gets to the target application, the PassTicket is passed to RACF where it undergoes the validation process using the PassTicket evaluation algorithm. While in transmission, a PassTicket is treated just like a password, which means that it is already supported by existing application and networking software. In addition, its short lifetime and built-in replay protection make it highly hacker-resistant when compared to the password. Since PassTicket technology makes use of existing sign-on protocols and does not require the reprogramming of existing host-centric networking applications, PassTickets are already finding use in applications such as the MVS Time Sharing Option (TSO), the MVS Information Management System (IMS), the Customer Information Control System (CICS), and in the MVS Advanced Program-to-Program Communication (MVS/APPC) environments.

The PassTicket approach may also be used to establish a sign-on environment, as illustrated in Figure 3, in which a user who is *already authenticated* to the network, may access an MVS host application without the need to be reauthenticated (or enter a password again). The request is sent to the network security server that generates a PassTicket and returns it to the user's client function at a workstation. At the workstation, the PassTicket is used by the client function in place of the user's MVS RACF password, meaning that the user need not enter the password to be reauthenticated. Note that in this scenario the strength of authentication, inherited from the use of a third-party network security server, is maintained by communicating a PassTicket to the host application instead of the traditional clear-text password.

The above scenario also demonstrates how a single sign-on environment can be achieved in a mixed environment of DCE client/server and host-based MVS systems with existing application servers. A single sign-on relieves end users from entering their passwords every time a service is requested from some other host machine in the network. As a result of an initial authentication, the end user's credentials are inherited among the network computing platforms, and transparently sent to the target machine without the end user's interference.

Controlling access to resources

In the last section we showed how the access control system, in concurrence with the Lampson model, causes a user security context to be generated when a user is properly authenticated and accesses the system. The user security context remains available so that the user can be identified during subsequent access control and audit processing. In this section, we focus on the resource access authorization checking process from both a theoretical and implementation viewpoint. Again, using RACF as the example, we discuss how access authorization checking is implemented in the MVS host-centric environment.

Background on resource authorization checking. Generally, access control to information resources can be classified into two broad categories: *discretionary* and *mandatory*.

Discretionary access control. Discretionary access control allows system users to grant or deny access as they choose to resources over which they have control. Users gain control over a resource if they create it, if they are the system's security administrator, or if some other user of the system has given propagation of access rights.

Mandatory access control. Mandatory access control is a way of restricting access to resources based on the sensitivity of the information contained within. To access a particular piece of information, one must hold the proper security clearance for that information and, more importantly, have a need to know. In accordance with the "read-down only" rule of a multilevel security system as defined by the Bell and La Padula model,[10] mandatory access control is enforced by assigning fixed security attributes, called *sensitivity labels*, to users and resources.

Users or *subjects* are able to access information only at a level of classification equal or lower than their sensitivity label, which is in effect a *clearance*, in accordance to a partial ordering relationship expressed using the term *dominance*. Users have no control over modifying or acquiring sensitivity labels that can be assigned to them only by proper organizational components responsible for resource sensitivity analyses and classification, and subject-clearance assignment. Sensitivity labels, also called security labels, can be multidimensional constructs, consisting of both hierarchical, linearly ordered, sensitivity levels (such as UNCLASSIFIED, CONFIDENTIAL, SECRET, and TOP SECRET), and nonhierarchical security categories, also called *compartments*, that can reflect the status and position of the subject (such as official rank and organizational component).

Mandatory versus discretionary. With mandatory access control, access rights are not susceptible to propagation among the population of users. In systems using both mandatory and discretionary access control, mandatory usually takes precedence in order to maintain the intended compartmentalization of subjects and information. To be effective, mandatory access control implementations require careful classification of users and information. This can translate into significant organizational overhead, resulting in an organization with very restricted information flow between its components. Mandatory access control has its roots in the military and related government agencies that have the need to implement compartmental security policies on the same computing system. To date, mandatory access control checking has not been widely adopted by commercial enterprise computing installations.

It is worth noting that the RACF support for mandatory access control has led to the evaluation of MVS 3.1.3 system by the National Computer Security Center at the B1 level of trust, as defined by the Department of Defense Computer System Evaluation Criteria.[11]

Discretionary access control, by its nature, allows an individual who has assess to information to decide

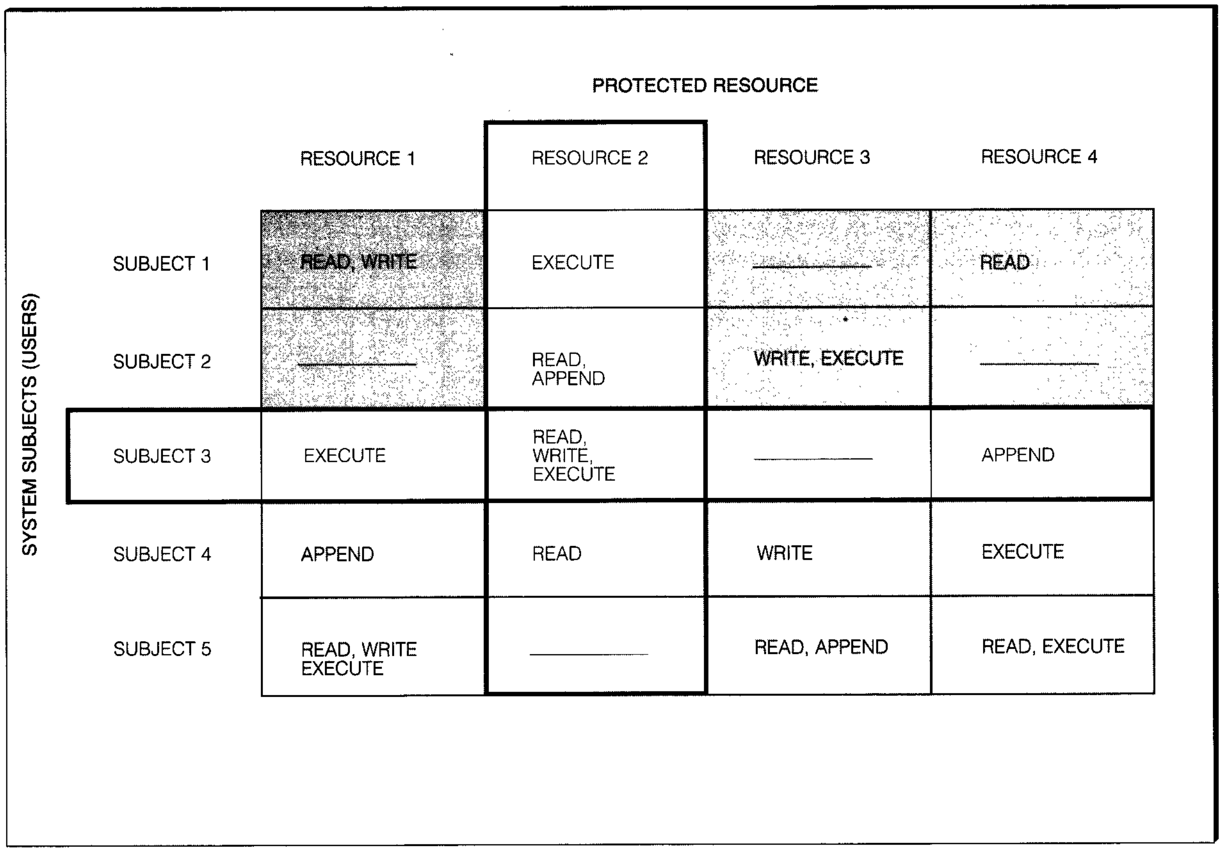


Figure 4 Access matrix modeling the protection of four resources

PROTECTED RESOURCE

SYSTEM SUBJECTS (USERS)

	RESOURCE 1	RESOURCE 2	RESOURCE 3	RESOURCE 4
SUBJECT 1	READ, WRITE	EXECUTE	————	READ
SUBJECT 2	————	READ, APPEND	WRITE, EXECUTE	————
SUBJECT 3	EXECUTE	READ, WRITE, EXECUTE	————	APPEND
SUBJECT 4	APPEND	READ	WRITE	EXECUTE
SUBJECT 5	READ, WRITE EXECUTE	————	READ, APPEND	READ, EXECUTE

who else within the organization should also have access, according to the dynamics of the business. Although discretionary access is the most common access control model used in commercial computing, it does not mean that one approach must be selected and used to the exclusion of the other. In fact, RACF on MVS allows installations to select a mandatory or discretionary access control implementation and various combinations of the two on the same MVS system. This kind of flexibility has proven to be necessary and should not be overlooked in the design of future access control systems.

Introduction to the access matrix model. As we have discussed in an earlier section, before access control software became available, file management systems employed a form of password protection to guard files from undesired access. The implementation of the Lampson reference monitor concept, and with it the notion of a user security context (or *authenticated identity*) for user processes, made possible much more efficient and effective access control logical mechanisms. In one manner or another, all of these, which we will discuss shortly, are examples of the *access matrix model.*[12]

Access matrix model theory. Theoretically speaking, the access matrix model is a variant of the finite state machine model, where each matrix corresponds to a state variable and the transition functions correspond to the processes of granting and revoking permissions to subjects that transform the matrix from one state to another. Figure 4 is an example of such an access matrix model. The rows of this matrix represent the system subjects while the columns correspond to the protected resources. This particular model illustrates the access control environment of five subjects to four resources; the intersection of a row and a column indicates the access mode a subject has to a corresponding resource.

Irrespective of the underlying security model used to enforce a security policy, the well-known safety problem of computer science remains unsolvable.[12] This problem seeks to determine, for a given subject in a state of any protection model, whether a particular access commensurate to a particular resource can be obtained. Although the difficulty in defining such a problem is clearly a theoretical limitation to the question of safety, it does not mean that for a given specific model the problem is not solvable. Rather, it only means there is no single algorithm that systematically solves the safety problem for all the instances of security models and policies.

Mechanisms that implement the access matrix model. Due to the sparsity of its matrix, the access matrix model has been mostly implemented in the form of either protection bits, a capability list, or an access control list (ACL).

The *protection bits* mechanism is seen mainly with UNIX systems. Here, only a few bits of access control information are attached to each resource. These bits indicate access modes for, usually, three classes of users: the owner of the resource, the user's group, and the rest of the system users. Each bit can be turned on or off according to an access policy that the owner chooses. The shortcoming of this mechanism is that one cannot uniquely assign an access mode to a specific user or group other than the owning user's group.

In the *capability list* approach to access control, the security system manages users rather than resources. Each user is assigned a list of capabilities enumerating the resources accessible by that user, along with the access types from a space of access rights with defined semantics. A capability list corresponds to a row of the access matrix; Figure 4 highlights a capability list for Subject 3. The task of managing access to resources becomes that of creating, updating, and deleting capability lists. While a capability list mechanism is effective in enumerating all the resources to which a user can have access, it has its own set of shortcomings. For the system to find out all the subjects having access to a particular resource, an exhaustive search of all the capability lists for all users in the system is required; a task that can be costly. Administering access controls is complicated since multiple capability lists have to be updated in order to grant or revoke access to a single resource. Computer Associates' CA-TopSecret is an example of a commercial system that is largely capability-based.

The converse of the capability list, and the most commonly used access control model implementation, is the *access control list*. In this implementation, security information for access control is associated with the resources instead of with the users, as we have seen with the capability list. A resource is assigned an access control list maintaining the subjects that can access the resource, along with the type of access for each subject. Figure 4 highlights an access control list for Resource 2. Frequent operations such as adding or deleting subjects from the access list of specific resources can be done efficiently. Likewise, a request to list all users that are allowed access to a particular resource can be satisfied quite simply by scanning the single access list for the resource.

In addition, the access control list model may lend itself better to the client/server distributed computing model. Since it is natural for the server to perform access control to its underlying resources, it becomes also natural for that server to maintain the security policy of its own resource. In other words, managing access control lists local to the server is inherent to the client/server model. Conversely, in the capability list mechanism, capabilities representing the user must be transmitted over the network to the remote server, which introduces concern over the integrity of the transmitted capability in addition to the issue of access privilege semantics as applications cross domains. An access privilege packed in some user's capability list and transmitted over to a remote server may be interpreted differently by that server. Note that when using an access-control-list-based authorization mechanism in a distributed environment, the scope of the semantics for access privileges becomes limited to the server, thus avoiding ambiguity. RACF is an example of an access control software package that implements the access-control-list-based access control model. Access control in DCE is also based on the access control list model using the POSIX type of list.

In real-world host-centric computing implementations, especially time-sharing installations, there will be many individual data files and other resources that are owned by individual users. Usually, the access control system will allow the owners of resources to access their own resources by default, that is, without specific administrative activity. End users normally do not have administrative authority over resources owned by other users or groups.

Matrix model access control implementation in RACF. Depending, to some extent, on the application, RACF employs several methods of matrix model access control: the use of permission bits, as in the OpenEdition POSIX support; capability lists, as in security categories; and access lists, the latter being the most prevalent. In the RACF implementation of access lists, all RACF protected resources fall into a *resource class* expanding on the access control list concept described above.

RACF access control resource classes. In the earliest version of RACF, the only resources protected were files residing on disk and tape storage. Over the years, enhancements to RACF and the security requirements of the MVS system have broadened the range of resources protected by RACF. The RACF approach to resource protection arranges resources into *classes* where each class identifies a set of similar resources such as files, tapes, and disk volumes. The MVS component that logically manages the resource (the resource manager) invokes RACF access authorization checking through the SAF interface, passing RACF a "handle" to the logical resource. This handle usually consists of the abstract name by which the resource is generally known within the computer system and the user population. RACF uses the name to "look up" whatever resource profiles RACF may have that pertain to this particular resource. The profile contains the security attributes of the resource, including the resource owner, and the *access control list* (the list of users and groups that are allowed access to the resource and their levels of access).

Advantage of separating the resource and the access list. This approach provides a level of abstraction between the actual resource and the rules that control access to it. This is in contrast to some other access control implementations in which the access control rules are physically linked with the actual resource, such as is the case with UNIX file permission bits that are attached to the resource. Separating the access control lists (within the resource profiles) from the resources allows the profiles to be stored on the RACF database where they can easily be used as data for reporting and auditing programs, which we will discuss in a later section. In addition, the storing of resource access control profiles and access control lists within the RACF database allows the administration and maintenance of the access lists to be the responsibility of the access control software rather than forcing this responsibility onto the applications that use the resource. This allows centralized administration of access control to be easily established as the default, while supporting decentralized administration by simply delegating the authority to administer particular sets of resource profiles as appropriate.

Resource profile owners. Each profile is attributed to an owner that could be either a RACF user or a RACF group. As we have mentioned earlier in our discussion of administration, the owner of a profile has administrative privileges over that profile. In addition, users who have been granted administrative privileges for a group have these administrative privileges over all resource profiles owned by that group, and any descending subgroups.

RACF separates the resource and the access list for centralization of access control.

Discrete or generic profiles. Each resource or a group of resources within a given class, can be protected by a resource profile that can be either *discrete* or *generic* in coverage. A discrete profile has a one-to-one relationship with the resource it protects; i.e., each profile is associated with one resource only, thus supporting a fine granularity of control. A generic profile, on the other hand, can apply to more than one resource within the same resource class that happens to share a common naming structure and common access control requirements. By using generic profiles, an installation can protect many resources with comparatively few access control profiles, therefore improving the efficiency of the access control administrative effort as well as the efficiency of the software access control system. Also, access privileges, established with generic profiles, transcend the actual existence of the particular resource; i.e., they can be established before the resource comes into existence and remain after it has gone away. This concept is important in that it allows a security policy to be implemented and essentially remain static, while allowing exception cases to be handled easily.

Other uses for the resource profile. Resource access control profiles can carry more than a simple access control list. RACF access control lists provide for a universal access privilege that indicates the type of access that any user not specifically indicated in the access list can have to the resource; it is the equivalent of UNIX *other*. An entry in the access list may correspond to a single user or to a RACF group of users. In addition, a security label can be retrieved from a resource profile by the access authorization procedure when mandatory access is active in the system.

Auditing criteria. In addition to maintaining access control information for the resource, a RACF resource profile can also maintain auditing control information related to the use of the resource. This control information establishes the conditions under which access attempts to resources are to be logged in the system audit trail. For instance, a user may elect to have automatically audited any failed Write attempts—because of an unsuccessful access control authorization check—to any of the user-owned files. Further, the user may identify a person to whom an immediate notice would be sent as a result of the attempted violation. Auditing criteria established in this manner results in the generation of an auditing record by the access control software. Normally, it does not require any special programming, for the specific purpose of generating the audit record, on the part of the application.

Combinatorial access permissions. Another important feature implemented within RACF access control lists is the concept of *combinatorial* access permissions. With this option, known as a *conditional access list*, access permissions present in an access control list can take effect based on the satisfaction of *other conditions* indicated in the access control list. The other conditions might include the method used by the user to achieve access to the system; i.e., the port of entry, or the program that the user is executing when attempting to access the resource. In the case of the former, for instance, a user may be granted access to the resource only when the port of entry is through the user's daily work location and not through a telephone network. In the latter case, access permissions could be established such that a user can access a file from the payroll database, for instance, only while the user is executing the *legitimate payroll program* and not the user's favorite editor. Note that this feature can be also used to limit the damage of a malicious access resulting from a virus program attempting to spread in the system.[13]

Access control auditing and reporting

In the section on administration of access control systems, we introduced the basic concepts that support access control system administration within the host-centric computing model. We described the registry implementation within MVS RACF for users and resources as expressed in the Lampson reference model. We discussed how the data elements of the RACF registry are exposed to human administrators of the access control system by human interface software functions. In this section, again using RACF as the example, we will introduce some important additions to the basic human interface functions.

Large and complex access control registries. A database of profiles can maintain information on thousands of users and resources. Access control lists may contain a large number of entries. Extensive security information may be stored in subsystem segments of user profiles, and users may belong to many different groups. The large amount of security, access control, and other user- and system-related data in a registry can complicate the task of monitoring the system. The complexity becomes especially apparent when the administrator or auditor wishes to view the summation of a number of small pieces of information that, although logically related, are spread around the registry with no direct way to retrieve them as a set.

Relational database technology offers the capability to represent computerized information in various customer selectable *views* and *tables*. When incorporated into a relational database, information may also be organized and presented to an interrogator in ways not supported by the standard human interfaces included with the access control software. However, supporting the performance requirements of the access control and auditing process exceeds the capability of relational database technology. The problem then becomes transforming the information present on the access control system native registry into a form usable by relational database products. The *Database Unload* utility program was developed to address this task in RACF. This utility program will transform the contents of all RACF registry profiles into a readable sequential file. In turn, the file can be used as input to a relational database system, such as DB2*. This is an approach that precludes the need for separate monitoring applications, which can be costly.

To illustrate the ease and the efficiency of this approach, we include a simple structured query language (SQL) query that shows how to retrieve all the resources explicitly accessible by user HOGGAR (the user listed in the access control list for the resource)

```
SELECT RESOURCE
  FROM RACF.TABLE
  WHERE USER = 'HOGGAR'
```

where it is assumed that all resource profiles are unloaded into RACF.TABLE.

Furthermore, this request can be satisfied for different access types or resource classes by simply modifying the same query statement. For instance the following SQL query statements allow us to quickly obtain a list of all of the files for which HOGGAR has READ access.

```
SELECT RESOURCE
  FROM RACF.TABLE
  WHERE USER = 'HOGGAR' AND
      CLASS = 'DATASET' AND ACCESS = 'READ'
```

Big and complex audit records files. Auditing security-relevant events is a fundamental function of system security and an area that has been addressed in the MVS RACF example of the host-centric computing model. RACF provides a wide scope of auditable events at a fine level of detail. Certain system events are always written to the system audit trail, such as the failure by a user to establish a logon session because of an incorrectly specified password. Other events are never logged due to their irrelevance to system security, such as an attempt to list the content of a resource profile. In addition to the capability of selecting events for logging based on the request of a designated auditor, or the resource owner, the auditor has the option to specify auditing at the individual user profile level. In this case all the user's actions are audited. Other audit selection options include the ability to audit access to a selected individual resource by auditing either failed or successful access attempts, and the ability to set various system-wide auditing options. Traditionally, MVS RACF auditing consisted of recording a wide range of security-related information in the system audit trail using the MVS System Management Facility (SMF). The SMF audit trail, however, is stored in a nonreadable internal format and in similarity to the access control information stored within the registry, there can be a large amount of data to process.

Not surprisingly, the same approach used for monitoring the RACF database has been applied to the SMF audit trail with another utility program called the RACF SMF *Data Unload* utility. With this utility program, SMF records that represent security-relevant events are transformed into a readable sequential table, which can then be loaded into a relational database system for complex queries. This facilitates a very flexible and dynamic way of reporting. By writing and modifying simple SQL queries, an auditor can quickly request information on different system subjects and resources by time, location, and many other parameters. Furthermore, a wide range of predicates that manipulate the values of different attributes from the relational table can be used for selection.

Let us assume that the SMF records are unloaded into table RACF.SMF. The following SQL statements are used to find any log records indicating successful WRITE access to file ACCOUNTS after 5:00 P.M. and before 7:00 A.M.—the period during which most of an enterprise's employees are off-site.

```
SELECT USERID
  FROM RACF.SMF
  WHERE RESOURCE = 'ACCOUNTS' AND
          CLASS = 'DATASET' AND
          RACCESS = 'WRITE' AND
          GACCESS = 'WRITE' AND
          TIME NOT BETWEEN '7:00' AND '17:00'
```

where RACCESS is the requested access and GACCESS is the granted access. This approach is a powerful and flexible way to satisfy an auditor's request.

Conclusion

In this paper, we have reviewed the Lampson reference monitor and the access control matrix, which are two important theoretical models on which modern computing access control systems are based. Using IBM's RACF as the primary example, we have demonstrated how these models have been applied to the host-centric computing model and we have pointed out some of the specific strengths of the access control and security implementation on MVS with RACF.

But, the world of computing is changing and the once dominant host-centric model is being displaced by a combination of the host-centric, the distributed, and the network-centric models. This is driving a requirement for the various implementations of the models to become capable of interoperation with one another. One of the most important objectives of operating system designers is to address problems exposed by the differences that exist between operating system platforms. The implementation of the reference monitor model in the open distributed computing model lies with the Open Software Foundation Distributed Computing Environment known as DCE. There are significant differences between the DCE approach and the host-centric approach as implemented with MVS and RACF. These differences are challenging software system vendors to design products with synergistic interoperation between the two, so that a single sign-on and common access control policy environment can be supported. To be successful in the marketplace, the product set will need to reflect the strengths of both approaches, while addressing the new requirements of the network-centric model.

Certainly, in our attempt to produce a paper of reasonable size, we have not completely covered this topic. There are a number of additional publications that we could recommend and the simplest way to point to them is to refer the reader to the IBM Security Architecture Guide.[14] This document suggests solutions to computing system security problems not limited to the host-centric computing model but extending across various computing platforms, including the distributed and networking models.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Computer Associates International, X/Open Co. Ltd., Open Software Foundation, Inc., the Institute of Electrical and Electronics Engineers, MacIntosh Laboratories Inc., Microsoft Corporation, Novell, Inc., or Massachusetts Institute of Technology.

Cited references and note

1. B. W. Lampson, "Protection," *Proceedings of the 5th Princeton Symposium on Information Sciences and Systems* (1971).
2. M. Gasser, *Building a Secure Computer System*, Van Nostrand Reinhold Co., Inc., New York (1988), pp. 57–72.
3. H. M. Deitel, *Operating Systems*, Addison-Wesley Publishing Co., New York (1900).
4. S. R. Ames, M. Gasser, and R. R. Schell, "Security Kernel Design and Implementation: An Introduction," *Computer* **16**, No. 7, 14–22 (1983).
5. *RACF General User's Guide*, SC28-1341, IBM Corporation (1994); available through IBM branch offices.
6. *RACF Security Administrator's Guide*, SC28-1340, IBM Corporation (1994); available through IBM branch offices.
7. J. G. Steiner, B. C. Neuman, and J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *Proceedings of the Usenix Conference* (February 1988), pp. 191–202.
8. B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Transactions on Computer Systems* **10**, No. 4, 265–310 (November 1992).
9. Secure-ID, Inc. has developed a device the size of a credit card providing this function.
10. D. E. Bell and L. J. LaPadula, *Secure Computer Systems: Mathematical Foundations and Model*, Mitre Corp., M74-244, Bedford, MA 01730 (1973).
11. *Department of Defense Trusted Computer Systems Evaluation Criteria*, DoD 5200.28-STD, National Computer Security Center, Linthicum, MD (December 1985).
12. M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in Operating Systems," *Communications of the ACM* **19**, No. 8, 461–471 (1976).
13. M. L. King, "Do We Have a Virus Problem with MVS Systems?," *Computer Security Journal* **5**, No. 2, 135–148 (1989).
14. *IBM Security Architecture: A Model for Securing Information Systems*, SC28-8135 (1993); available through IBM branch offices.

Accepted for publication September 7, 1995.

Messaoud Benantar *IBM S/390 Division, 522 South Rd., Poughkeepsie, New York 12601 (electronic mail: benantar@vnet.ibm.com).* Dr. Benantar is an advisory programmer with the RACF/MVS development group in Poughkeepsie. He received his diplome d'ingenieur in computer science from the Université des Sciences et de Technologie, in Algiers, Algeria, in 1983, and an M.S. in 1986 and a Ph.D. in 1992, both in computer science, from Rensselaer Polytechnic Institute in Troy, New York. He subsequently joined IBM at the Poughkeepsie MVS development laboratory. His interests include system and network security, object-oriented computing systems, and parallel algorithms.

Rich Guski *IBM Corporation, 522 South Road, Poughkeepsie, New York 12601-5400 (electronic mail: guski@vnet.ibm.com).* Mr. Guski began his career in 1971 with the Mountain Bell Telephone Company in Denver, Colorado. He held assignments in computer operations, applications programming, and data security technical support. Mr. Guski joined IBM's Enterprise Systems Division in 1981 in Poughkeepsie, New York, as a programmer in RACF development, where he participated in the design and development of many versions of RACF. He is now a senior programmer in the RACF Design group. His responsibilities include determining the strategic direction for the RACF product and interfacing with customers and other IBM programming development laboratories. In addition, Mr. Guski is IBM's representative to the GUIDE Security Software project. He received a B.S. in computer technology from the New York Institute of Technology in 1971.

Kathleen M. Troidle *IBM Corporation, 1133 Westchester Avenue, White Plains, New York 10604 (electronic mail: ktroidle@vnet.ibm.com).* Ms. Troidle is a marketing support administrator with IBM's United States Sales and Services organization and has national product marketing responsibility for S/390 security and systems management software. As a security marketing specialist with IBM, she has been engaged in the various aspects of systems management from the mainframe to the workstation. She has helped many customers across the United States implement and use RACF as part of their overall security strategy, as well as helping to design comprehensive security solutions. As a systems analyst focused on security for the United States Air Force, and as a systems engineer and marketing representative with IBM, Ms. Troidle has been working in computer systems management since 1983. She has lectured across the United States on enterprise security, including annual RACF conferences, GUIDE, Enterprise-Wide Security conferences, and the first IBM International Cryptographic Symposium. She holds a B.A. in mathematics from The College of the Holy Cross (1983) and a master's in computer systems management from Creighton University (1987).

Reprint Order No. G321-5596.