The complexity of necessary administrative controls for computer service exceeds the capabilities of clerical methods. This paper presents a practical method for describing the external administrative environment in a data base which can be used by the operating system for dynamic enforcement of limits. An attempt is made to address consistently the different forms of data processing that may be concurrent in a single installation; included are general purpose time sharing, transaction oriented computing, and scientific computing. Described is the architecture of an operating system component that could be regarded as the interface between administrative security mechanisms and the security features of the system software.

Running prototypes exist. The long-range intention is to streamline such computer facility management functions as controlling access to specific services, processing power, and storage space; controlling access to the system data base; and gathering statistics needed for planning. Convenience to users is not degraded by the security mechanisms, but in fact is enhanced.

Administrative control of computing service

by H. M. Gladney

Corporate general managers, corporate comptrollers, and data processing managers are finding that their obligations as stewards of resources are being interpreted far more widely now than even in the quite recent past, and that the technical opportunities for willful or accidental misappropriation of resources seem to be expanding.[1] As the number of computer applications grows, the number of users, the number of terminals, and the number of data items will grow rapidly. The number and complexity of the related administrative decisions can be expected to grow even more rapidly and eventually become a constraint to growth.

Part of the challenge can be met by replacing clerical procedures with automatic procedures wherever possible in the computer installation and by making practical a great deal of delegation of resource administration. The problem is intimately related to questions of information protection and computer facility security. The mechanisms employed must, as a minimum, be consistent with protection objectives and can, in fact, provide one element of a comprehensive set of software security measures.

A start toward the required function was made in IBM's Research Division in 1970. A 1975 paper[2] described a set of operating-system extensions called the Installation Management Facility (IMF). IMF was intended to provide controlled access to system processing power and functions, controlled access to the system data base, decentralized authorization responsibility, improved transparency to the physical location of data, and numerous enhancements to the administrative interface with users. Experimental versions of IMF were implemented for OS/MVT (IBM's Operating System for Multiprogramming with a Variable number of Tasks) and for OS/SVS (Single Virtual Storage—that is, OS/VS2 Release 1). Based in part on these ideas and prototypes, an IBM product called the Resource Access Control Facility (RACF)[3] was developed. It is available for OS/MVS (the Operating System for Multiple Virtual Storage). This paper attempts an abstraction of the IMF architecture based on research done in other laboratories since 1972 and explores possible extensions.

Although the paper can be viewed as a specific proposal for new system function, the primary intent is to evolve a uniform, comprehensive external architecture that corresponds so closely to natural requirements that it is obviously "correct." Such an architecture is best expressed primarily in the definition of entities and their interrelationships. If these definitions are well chosen and complete, the system action desirable in any circumstance will be readily apparent, and system design can proceed. Much of the discussion uses, as examples, the functions and nomenclature of OS/MVS and of some subsystems available on an OS/MVS base. Where a specific mechanism is suggested, the intent is primarily to demonstrate that the proposal is feasible.

The title of the paper requires explanation. IMF is not described as a security mechanism because it alone cannot prevent illegitimate access and because such a description would limit the conception of what might be accomplished. In conjunction with other measures, IMF can confer a measure of security by providing readily available tabulations of intended access limitations. However, it is preferable to think of IMF as a mechanism that helps a data processing installation administer the interrelationships of users and valuable resources. This makes it possible to maintain and

administer access to resources in a manner tailored to each user's needs, and simultaneously to meet management requirements for control and audit.

history and background

IMF can be traced back to experiments in Project MAC at MIT.[4] Development of the ideas has continued independently in IBM Research[2] and at MIT[5] since about 1970 and has been paralleled at Cambridge University by Wilkes,[6] who was influenced by visits to MIT. Similar lines of investigation have been pursued at Boeing[7] and the University of Chicago.[8]

In the interest of brevity, it is presumed that the reader is familiar in detail with Reference 2, which describes the basic ideas of IMF. Additional necessary background information is provided by Saltzer and Schroeder[9] in an outstanding tutorial on software aspects of information protection, and in Wilkes' book[10] on the architecture of time sharing systems. In addition, familiarity with the external behavior and design of OS/VS is presumed.

Decentralized dynamic control of commodity resources, such as processor time and secondary storage occupancy, is the subject of separate treatment.[11] The intended use of the IMF data base as a catalog for an automatic secondary storage hierarchy did not materialize; instead, an MVS automatic storage hierarchy evolved separately and was the inspiration for a recently announced Hierarchical Storage Manager.[12] There is still the potential for integration of the underlying data catalogs.

basic mechanism

The basic mechanism of IMF is simple. There exists a single inventory which describes subjects and data objects and their interrelationships. The entry for a protected object contains a list of authorized users. When a session is initiated, the system uses IMF data and other mechanisms to verify that the user is who he claims to be, and it creates a protected control block called the access control environment block, or ACEB, which summarizes the user's identification data. Later, when access to a protected resource is requested, the ACEB contents are compared with an inventory entry for the resource in question to determine whether the type of access requested should be permitted.

The RACF implementation[3] provides a set of supervisor calls to check, manipulate, and maintain the inventory contents. Each subsystem is responsible for using these interfaces to provide access control. RACF, in turn, uses more basic operating system mechanisms and depends on their integrity.

summary of the paper

Definition of administrative control mechanisms must proceed at several levels. IMF can be considered to be primarily a definition of entities and their relationships. The value of such definitions depends on how closely they correspond to the user's intuitive

notions, on whether they are mutually consistent, and on the extent to which they are complete over the full range of services offered and resources to be managed.

Data structures, the next level of definition, provide much more detail and precision about the subjects and protected objects than the naked definitions, at the expense of losing some generality. These data structures are simpler than the system commands and procedures which access and modify them, and therefore provide a basis from which command and procedure definition follows naturally.

In this paper, emphasis is on architectural definitions and examination of external function that might be desirable. Secondarily, the supporting data structures are defined in sufficient detail to demonstrate the feasibility of the scheme. The next levels, command structures and internal interfaces, receive very little attention. Described in detail in Reference 13, they are primarily of specialized interest, illustrating precisely how authority can be propagated in a decentralized environment. Finally, the paper rarely mentions where and how the operating system uses the interfaces to the authority tables because use of the interfaces is largely obvious in OS/VS. It is the subject of much design research on the central portions of operating systems and therefore is susceptible to obsolescence.

key terms

Precise definitions of the terms listed below are essential in describing the architecture of IMF. It is the elaboration of the implications of these definitions that gives rise to the rest of the described structure. More complete glossaries are provided in References 9 and 11.

Account—an *inventory* entry for accumulating system usage measures for financial purposes and for tabulating related authorities.

Author—see *owner*.

Authority—see *data access authority, group authority*.

Connect entry—a relationship between a *user* and a *group*, stored in a unique inventory record. Each user is related to at least one group. A user can be actively connected to only one group at any instant.

Data access authority—every protected data set has an identified *owner*, who may be a user or a group. The kinds of access authority that can be granted by an owner to another *principal* are:

- ACCESS CONTROL LIST—the authority to amend the privileges of other users of the object. A user with this privilege is an owner.

- ALTER—the authority to read, write, erase, rename, or execute a data set, but not to amend the access list.
- APPEND—the authority to open a data set only for extension.
- EXECUTE—the authority to open a data set only to load a module for execution.
- NO—the specified principal has no authority to open a given data set for any purpose.
- READ—the authority to open a data set for input.
- READ/WRITE—the authority to open a data set for input, output, or updating. READ/WRITE includes APPEND.
- WRITE—the authority to open a data set for output. WRITE does not include APPEND, but only update-in-place.

Group—a set of users or other groups identified for administrative purposes. Each group has at least one user with authorized access to resources granted to that group. Every group has a superior group.

Group authority—each principal connected to a group has specific authority relative to the group resources. The possibilities are:

- CONTROL—the authority to permit users already connected to the system to join the specified group. CONTROL implies all the privileges of CREATE.
- CREATE—the authority to create permanent data sets owned at the group level. CREATE implies all the privilege of USE.
- JOIN—the authority to add new users to the system with any group privilege. JOIN also allows a user to define new subgroups. This authority implies all the privileges of CONTROL. The author of a group is automatically presumed to have JOIN authority for the group.
- RUN—the authority to use the system under group sponsorship and to create temporary data sets.
- USE—the authority to create permanent data sets and to use the system under control of account numbers explicitly assigned by the group administrator.

Inventory—the complete collection of data required to administer access to system resources, including descriptions of protected objects and of potential accessors and statistics about past use of resources.

Owner, author—the owner of a resource is any principal who has the right and responsibility to define access privileges for that resource. The user who creates a resource is referred to as its *author*. The author is always an owner.

Principal—in general, the entity in a computer system to which authorizations are granted. Users and groups are examples. Spe-

cifically, a principal is a Boolean construct of ascertainable attributes about a user and the environment in which privileges are requested.

Security officer—an official charged with enforcing, improving, and auditing security practices. The security officer can read most inventory contents but modify none.

Universal access field—a field in every access control list that describes the privileges of all principals not named explicitly or included in a named class.

User—a person who uses computing services. He is assigned an identifier called his USERID, which must be presented to gain access to the system. Listed below are special categories of users:

- **Administrative user**—one who is authorized to query and modify certain inventory records and fields not accessible to other users. The purpose is to facilitate accurate maintenance of accounting and other information required to manage the facility.
- **Master user**—one who is authorized to control the function of a specific subsystem.
- **Operations user**—one for whom selected authorization checking is bypassed so that certain maintenance functions, such as copying entire storage volumes, can be carried out.
- **Special user**—one for whom all access authority checks are bypassed, except for those in the sign-on procedure.
- **Supervisory user**—one who is authorized to control the function available to the users of a specific cluster of terminals.

abstract structure

There are two general categories of mechanisms for the control of sharing: ticket oriented and access-list oriented.[9] In each case, collections must be maintained. The crucial distinction is whether the collections are maintained by subjects or in connection with objects. In a ticket oriented mechanism, each subject must keep a collection of identifiers or keys for the objects of interest. Such keys are issued by the owners of protected objects and may or may not be amenable to replication by their holders. A collection of house keys is the most familiar example; each key might open several otherwise unrelated doors. In an access-list oriented mechanism, the permission controller for each protected object includes or has access to a tabulation of authorized subjects, with an indication of the type or types of access to which each subject is authorized. There also must be a means of authenticating the claimed identity of each subject—either a unique characteristic of the subject, such as a fingerprint, or a ticket specific to that subject, such as a personal password or an identity badge. Access lists are often implicit. An executive secretary limiting access to his principal, for example, uses an implicit access list, granting

different levels of access to the principal's superior, spouse, subordinates, and customers, and identifying and authenticating each by known personal characteristics.

Both mechanisms are useful in operating systems. The list oriented mechanism requires that a gatekeeper search his tabulation every time access is requested and verify that all conditions of access are met; the subject need only maintain and present a single piece of information—his authenticatable identification. The ticket mechanism requires each subject to save and protect the set of keys required and present the correct key whenever a privilege is requested. Clearly, access lists are more convenient for subjects, and tickets for gatekeepers. List oriented systems have a property not shared by ticket oriented systems: they permit revocation of privilege without consultation with the potential subject. Also, with list orientation, it is easy for an auditor to determine the range of access without himself having the access.

In either case, there must be human communication outside the mechanical system. For list orientation, the requestor must communicate his identity to the resource owner or the gatekeeper. For ticket orientation, the owner must transfer a key to the requestor, who might transfer it to further subjects.

It is a central notion that every protected object has an explicitly identified owner. Also, the description of every subject and every statement of interrelationship of entities are protected objects and therefore have owners, commonly referred to as authors.

IMF can be regarded as a set of mechanisms for converting external authentication keys into access-list requests and subsequently into keys for specific resources. It also includes the means for creating and maintaining access lists. When a user requests service, IMF supports a dialog that creates an internal table representing an authenticated identification, together with all circumstances potentially pertinent to subsequent requests. Whenever a resource request occurs, this identification is checked against an access list, and tickets are created to allow use of the resource. In OS/VS, tickets are generally represented by valid control blocks, whose creation is dependent on many checks in addition to those peculiar to the user. Such control blocks are short-lived compared to the access lists. For IMF to be effective as a security aid, it must not be possible to create, modify, destroy, or read these blocks except under certain clearly specified circumstances. Delineation of an adequate set of circumstances, and creation of a system kernel that demonstrably implements these rules, are problems currently receiving much attention.[9, 14, 15]

There are three categories of rules for granting access: explicit or propagated permission granted by the resource owner to specific

Table 1 Candidates for administrative control

Structured resources
Data sets
Terminals
Accounts
Groups
Storage volumes
Transactions executions
Program library members
Services
Each application subsystem
Batch service classes
Security classifications
Administrative status (e.g., *operator*)
Commodities
Processor time
Processor time weighted according to time of day
Priority service processor time
Storage space
Session elapsed time

sets of subjects, relationships between the security level of the protected object and the current administrative privilege of the requestor, and external circumstances of the request (for example, some bank vaults cannot be opened at night). Examples of administrative privileges are military security clearance levels and such computing facility functions as *operator* and *security officer*. Commonly several criteria apply concurrently, so that all three types of mechanism must be accommodated without conflict in any general system.

Every resource of a computing service can be classified in one of three categories: structured resources, service resources, and commodities. There are many structured resources in a system, each with different internal characteristics, different ownership and access requirements, and different usage patterns. Access to a structured resource generally is not determined by the organizational hierarchy; for example, the manager of an owner is not automatically granted the owner's privileges. Structured resources can also be subjects, as when access is granted to a program that is a member of a protected data set. A service resource is simply the privilege to use a service class, possibly with circumstantial limitations. Only one example of each service resource exists in a system; it is used by many parties, and either hierarchical or centralized administrative propagation of privilege may be appropriate. A commodity resource is measured quantitatively and might be a candidate for accounting and billing. Generally, use of a commodity resource is associated with either a structured or service resource; for example, storage space with a data set, and processor time with APL service. Hierarchical propagation of commodity resources is always appropriate.

IMF treats each class of resource differently. Structured resources are represented by individual entries in the inventory. (Closely related sets of structured resources might be represented by single entries.) For each instance, the system automatically maintains a description and usage statistics, and provides the owner or owners with the means of specifying access privileges. Service privileges are maintained as simple *yes* or *no* entries in subject inventory entries and require only simple checks at sign-on time. Commodity resource allocations are maintained in inventory subject entries and are propagated hierarchically with the aid of system commands. As commodities are used up, the system maintains records automatically, and as limits are reached, the system scheduling components are used to deny or defer service.[11] Table 1 lists examples of each resource type.

Inventory structure and contents

In the early prototype,[2] the inventory was a single data set which also served as the system catalog. In RACF,[3] the inventory is a

similar data set which does not include records for unprotected data sets. It would not markedly complicate implementation, in fact, if each distinct class of inventory entry were stored in a distinct data set. Only for data-describing entries are more complex choices available. These entries can be stored as system directories, such as OS/VS catalogs, or in distinct data sets which are searched only for protected objects, or they can be distributed together with the protected objects, possibly also serving directory functions.[9, 16] The first two alternatives simplify centralized administrative functions. However, the specific storage structure and record formats of the inventory are, for the most part, irrelevant to this paper and will not be further discussed.

The principal contents of many of the currently defined inventory entry types are summarized in Tables 2 through 9.

Access control and auditing

Access control lists as previously described[2] do not provide all the flexibility required for some applications. For example in a commercial installation in which the person running a job neither prepared the programs nor owns the data bases, it may be desirable to restrict his access to each data set to occasions when he is using only a certain set of program libraries. He may be further restricted in having access to the data base only from certain terminals at certain times.

IMF grants access to individual users more or less independently of the program being executed—a somewhat different approach than that discussed by Saltzer and Schroeder,[9] in which authority is granted to a process to which a user has been admitted. Authority based (in part) on the identity of the program being executed poses a problem in OS/VS. Between the time when the module is identified and the time when access is exploited, the module in control may have changed. While it might be possible to know the complete set of programs that could gain control from the one identified, it seems extremely complex to ensure that an unacceptable program is not masquerading as one of them. The solution to the problem of adequate identification of the program in control is left to other efforts. However, in anticipation of progress, the necessary data and command structures in the external interface must be defined.

There is also provision for permanent records of unauthorized access attempts and such other resource accesses as the installation or individual owners specify. The recommended collection data sets are the accounting logs; however, individual streams of data-set, sign-on, terminal, and transaction records may be directed to other data sets. Violation descriptions can also be directed to a security officer's terminal.

Table 2 Contents of inventory *data set* entry

Base portion, volume, association, and password fields
As defined for VSAM catalogs
Statistics fields
As defined for VSAM catalogs, with the following additions:
Dates of latest reference, change, backup, and migration or staging
Numbers of OPENs for input and output
Control information for migration, staging, and backup
Access control fields
Pointer to owner of security verification routine
Universal access flags
Audit trail flags
Document security level
Names and privileges of authorized principals
Change-tracking fields
Name of tracking data set
Tracking action request flags and retention rules
Event types to be tracked
Member names for special treatment with parameters as above
Date of latest change of each named member

The entry for private storage volumes (primarily tapes) would be quite similar.

access control rules

An access control list may be associated with every named protected object, at the owner's option. For each class of protected resource, it is possible to associate an installation security verification routine. It might also be desirable to allow the owner of each object to define and interpose an owner security verification routine, but the current architecture of OS/VS does not readily provide for storage and cataloging of such routines. The protection options in effect are specified by the owner of the protected object, subject to minimal installation standards.

In an access control list, the priority of checking is as follows: principal, USERID, current connect group, universal access. As soon as a name match occurs, the privilege indicated is assigned and there is no further searching; hence a user may be denied access even though access is granted to other members of his group. The grouping of users tends to keep access lists short.

In the early prototype,[2] the access rules for data were hierarchical; that is, WRITE implied READ. This design has been abandoned, since there are situations in which keyed insertions might not require a previous READ. As another example, it is quite reasonable that a user be permitted to extend a data set he is authorized to neither read nor update.

A user with the *special* attribute is exempt from access control checking, except for sign-on processing. For such a user, and also for users with the *master, supervisory, operations*, and *administrative* attributes, terminal identification support may be used to limit access to relatively secure terminals.

One bit in the user entry is a REVOKE flag. If it is set, the user is not permitted to sign on. It can be set only by a JOIN user in the user's default group or a superior group, or by an administrative user. It can be reset only by an administrative user.

A user with the *security officer* attribute can inspect the contents of any inventory entry, with the exception of password fields, but he does not have access to the protected object itself. For example, he can ascertain who is permitted to use a specific terminal that he himself is not authorized to use. He cannot alter any inventory information.

Table 3 Contents of inventory *group* entry

Base portion

Group name
Superior group name
Author, creation and expiration dates
Default protection for new data sets
Default audit trail action
Default security code for new data

Access control fields

Names and privileges of authorized principals

Subgroup fields

Names and defined subgroups

Account number fields

Control flags for administrative users
Names of valid accounts

Statistics fields

Date group entry last used
Limits and statistics for each defined commodity resource

Privileges fields (one set for each service available)

Service name and service subset name (e.g., APL)
Control flags (meaning depends on service type)
Control parameters (e.g., maximal address space)
Installation-defined fields

The connect entry is similar except for the base portion, which contains both a USERID and a group name.

definition of a principal

A principal is the entity to which authorization is granted. There are situations in which a one-to-one correspondence of users to principals is not adequate. Stepczyk[17] summarizes the general access rule as *subject verb object environment*; if this sequence is reordered as *subject in-environment verb object*, it is a good model of the IMF mechanism.

A principal is represented by an inventory entry, as described in Table 6. The elements to be checked can be system-state vari-

ables, values that occur in the current access control environment block (ACEB), global-state variables such as the time of day, and other inventory contents. In the last case, it is not necessary that the claimant be authorized to read the referenced inventory entry, since the checking can be done in a system key inaccessible to him.

Manipulation of principal entries is supported by a PRINICPAL command and subcommands, which define a Boolean expression to be evaluated when a claim is issued. (For detailed discussion of these commands and subcommands, see Reference 13.) Items can be entered in any order, and a definition need not be completed in a single terminal session. The principal entry may contain expressions that do not contribute to the evaluation; they are simply ignored. It is an installation option whether missing expressions are assumed to take the value *false* or *true*. *False* is recommended. The use of each principal entry is controlled by an access list similar to that for a data-set entry.

Saltzer and Schroeder[9] suggest the importance of situations in which a data base is to be modified only if a committee agrees, much as entry to a safety deposit box is controlled in a bank. One way of handling the requirement that two users concur is that each would have a secondary password which he would communicate to the user who wished to cooperate with him as a dual principal. In the PRINCIPAL command, a CONCUR operand designates a list of users from which two must concur before either's claim is accepted. When the claim is issued, the issuer is prompted to supply the name and secondary password of his partner.

Identification as a named principal is claimed explicitly with the sequence CLAIM (PRINCIPAL-NAME). For application programmers, a macroinstruction and corresponding high-level-language subroutine calls are defined. No password is required as part of a claim (but one can be used if the extra protection, and annoyance, are judged worthwhile). A principal claim is rejected if the issuer does not have READ authority for the inventory principal entry.

Table 4 Contents of inventory *user* entry

Base portion
User identification[a]
Default group name[b]
Author, creation and expiration dates[a]
Password and secondary password[b]
Password change interval, last change date[a]
Special privileges available (e.g., administrative user)[a]
Date and time of latest job
Date and time of latest terminal session
Data protection profile
Default protection for new data[c]
Default audit trail action[c]
Default security code for new data[c]
Default data retention period[b]
Default catalog for new data sets[b]
User description
User name, employee number, division, location[d]
Mailing address and telephone number[b]
Department name and number[b]
Service profile
One for each service available[b]

[a]Suitable superior privilege is required to change these fields.
[b]The user has authority to control these fields.
[c]Either the user or the author of the user entry can change these fields.
[d]The user or the author can set these fields, and the administrative user can reset them. The administrative user can set a flag preventing further modification except by an administrative user.

Subsystem support

From the point of view of IMF, each type of computing service is represented by a defined subsystem to which sets of users have authorized access. Even batch and general purpose terminal services are represented by subsystems, although this structure is required primarily for application subsystems. IMF is intended not to compete with or supplant access control mechanisms in subsystems, but to supplement them by providing interfaces to resources not contained entirely within the subsystems and to user

Table 5 Contents of inventory *terminal* entry

Base portion

Terminal name (may include node name in a sibling network node)
Device type
Author, creation and expiration dates
Hardware security features installed

Access control fields

Names and privileges of principals
Audit trail action required
Maximal security level permitted
Control limits for sign-on
Control limits for session
Security level as a function of time of day
Special security action (e.g., *lock keyboard on violation X*)
Limits of access by special users (e.g., *administrative user not permitted*)

Statistics fields

Dates of latest change and latest use
Number of valid and denied sign-ons
Total terminal session time

The *relator* entry differs primarily in including the name of a single authorized principal and in limiting and recording only his actions.

and principal definitions. In some cases, IMF may provide a more flexible or convenient method of specifying access lists than is currently supported in the subsystem. A major motivation is to provide an access control system that is compatible across all subsystems because users frequently have authorized access to several subsystems. This goal is compatible with the objectives and architecture of modern message control programs such as IBM's Virtual Telecommunications Access Method (VTAM).[18]

The sign-on process can be similar for all subsystems. The user selects a particular subsystem by choosing an entry port or, if a single message control program supports multiple subsystems, by specifying the subsystem name as part of the log-on procedure. His USERID identifies him to the system. The user can choose among the attributes for which he has options. These attributes may include the group to which he is connected, the account to which the session is to be charged, and the administrative privilege levels that should be available for the session. As part of the process, the user is prompted for authentication by password or mechanical device.

A message is issued and a violation record is added to the system audit trail if the sign-on is unsuccessful for any reason. (Successful sign-ons are of course also logged as part of the installation accounting procedure.) A product of the sign-on process is the previously mentioned ACEB, the principal contents of which are listed in Table 10.

Data access control at the record or field level is a subsystem responsibility; by limiting users to specific transactions, they may be limited to subsets of the data base. In the ACEB, IMF attempts to describe the environment in which a transaction is requested in sufficient detail so that rules for access to particular records or fields can be enforced by the subsystem for a user who does not have permission to use a data set except via the subsystem. The prototype mechanism is:

- The application subsystem opens a data set after issuing a claim as an adequately privileged principal.
- The subsystem limits the use of its own transactions to users based on inventory lists.
- Access lists for the subsystem and transaction use, for the subsystem code, and for the data-base elements (data sets) can be maintained separately. In each case, the person responsible is clearly identified.
- Construction of all the access lists is supported by IMF commands which can be issued from within the subsystem.

Each subsystem may have an audit-trail data set for recording possible security violations.

For each distinct subsystem, there must be an inventory subsystem entry that contains the names of all groups directly authorized to use the subsystem. (If the root node of the group tree is included, the subsystem entry does not restrict the use of the named service.) Access to a subsystem entry is managed as for a data set, except that the default options restrict access severely. Only a user with the *administrative* attribute can be the author of a new subsystem entry.

IMF also maintains subsystem access and statistics information in group and connect entries. The rules for authorization propagation are similar to those for account-number propagation described below. Whether this support is used for installation-defined subsystems is an installation option. Some subsystems, such as the Customer Information Control System (CICS),[19] define certain control functions which should be available only at specially controlled terminals called master or supervisory terminals. The IMF approach is slightly different and more flexible: a user can be assigned the master and supervisory functions. Then if he signs on to an authorized terminal claiming the special attribute, he will be permitted to use the defined functions.

IMF also can maintain more detailed lists which indicate authority for access to specifically named subsystem services and control of transaction journals. This feature is intended primarily to support transaction oriented data base/data communications application subsystems. A transaction entry can be specified independently for each transaction or named class of transactions. A transaction not named in the subsystem transaction list does not enjoy any protection beyond that afforded the subsystem.

For three types of subsystems—batch, general purpose interactive as represented by IBM's Time Sharing Option (TSO), and interactive scientific as represented by APL—the IMF support already has been described.[2] In addition, some functional extensions have been defined.[13]

IBM's Information Management System (IMS) is a suitable example of a transaction oriented data-base subsystem. From the point of view of IMF, the following circumstances are pertinent:

- Normal user access to the data base is by way of procedures called transactions. Centralized managerial control of the transaction logic is feasible and common.
- Differential control of access to individual records and fields in records in the data base commonly is required for confidentiality and privacy, as well as to reduce exposure to fraudulent modification.
- The system support staff—programmers, data-base administrators, security officers—and the end users are generally different persons in different departments.

Table 6 Contents of inventory *principal* entry

Base portion

Principal identifier
Author, creation and termination dates
Secondary password and date last changed

Password fields

Similar to corresponding fields in *data set* entry

Access control fields

Similar to corresponding fields in *data set* entry; some field values have no meaning, however, and are ignored by the service routines

Set inclusion fields

Number of inclusions
For each inclusion:
  Inclusion name
  Variable to be checked[a]
  Arithmetic or logical operator
  Value list

Boolean fields

Number of Boolean expressions
For each expression:
  The expression name
  AND or OR
  List of named expressions

[a]Examples are: time of day, day of week, date, condition of access to some other named object, current or potential administrative privilege, name of current terminal, storage key.

transaction oriented subsystems

- Because of response-time requirements, the overhead for access-list resolution for each transaction may be insupportable. A ticket oriented approach to security at a fairly high level in the subsystem support code is commonly adopted.

The IMS control programs contain sensitive code and are intended to be run as OS/VS2 authorized programs.[20] IMS is started as an OS/VS system task at the system console. The data bases and lines are opened when IMS is initialized. The terminal operator does not have to sign on.

The protected objects recognized by the IMS security feature are transactions, IMS commands, program status vectors, data-base status vectors, and physical or logical terminals. Access to protected objects is provided by passwords, which must be issued by operators as part of terminal transactions; that is, ticket orientation is visible to the end user. Internally, the control program maintains a series of access matrices that limit the use of protected objects to specific terminals or require that a password be issued as part of each transaction or command.

IMS/VS[21] already has quite extensive features for limiting the privileges of individual terminal operators. There are many ways in which the IMF-IMS interface can be built; discussed below are two extreme alternatives.

first alternative—minimal subsystem modification

IMS can be treated as a protected subsystem, for which IMF is used to isolate data bases, terminals, and users from other applications to the greatest degree possible. The only logic modification required in this case is that the IMS control program issue an IMF macroinstruction to create an ACEB, followed by a CLAIM macroinstruction to identify itself as a principal to whom terminal and data-set access will be granted. Terminals and communications lines can be protected not only by being bound to the IMS message control program, but also by the terminal identification feature of IMF. Access to the data bases is authorized for IMS by granting permission to the previously mentioned principal, and also to IMS support personnel as required. The end user will see no change to IMS as previously defined. Nor is any change required to IMS security procedures.

second alternative—maximal support

A more radical approach is to discard the IMS disk-resident resource access library and rely on the inventory to store access tables, and on IMF commands to maintain them. During IMS initialization, as each data base and communications line is opened, the IMF interface in the OPEN routines checks for a principal identification match. Then IMS reads its IMF subsystem entry to determine which transactions, commands, program status vectors, and data-base status vectors are protected. For each transaction represented in the subsystem entry, the inventory transaction entry

is used to build the IMS security matrices. However, passwords in these matrices are regarded as USERIDs.

The control program will accept no commands or transactions from any terminal until a user is signed on. The privileges with which a user may sign on are described in his connect entry. Included are indicators of administrative privilege relative to IMS, maximal processing priority, minimal audit trail behavior, and minimal security level for output. The USERID is associated with the terminal identifier until the user signs off.

The end user will perceive several changes. Terminal security procedures will become more nearly similar to those for TSO; in particular, the user will be required to log on but not to issue a password with each transaction. Decentralized access control permits the use of local security officers and administrators, who should be more responsive than a central system administrator.

For installation management, the merits of this approach are that administrative burdens can be reduced; the security officer can concentrate on testing and auditing instead of administering tedious detail; and responsibility and accountability will be easier to trace to individuals than is currently the case.

data-base oriented subsystems

The foregoing approach evades the problems involved in protecting individual records and fields of a data base, at the cost of severely limiting the use that can be made of the data. The difficulties of having data access control dependent on both content and inquiry content are illustrated in articles such as that by Hsiao and Baum.[22] Currently it is not known how to specify access procedures simply enough for general use. Also, implementations of most proposed general schemes promise to be impossibly expensive because the attributes and relationships of any data item are much richer in information than the item itself.

These problems go far beyond access control and have stimulated investigation into relational data-base systems.[23] In the context of research by means of a prototype called System/R,[24] an access mechanism has been defined for a relational data-base system.[25] Although this work proceeded independently of the IMF work, it has strong similarities. From the point of view of IMF, the most pertinent characteristics of System/R authorization are:

- An objective of relational systems is to avoid constraints on the complexity of views of data. Access constraints are inherently part of such views, so that it may be undesirable to extract the access control mechanism as a semi-independent component.
- A high-level query language is included to avoid having application programmers who are distinct from end users.

Table 7 Contents of inventory *account* entry

Base portion
Name or number of account
Author, creation and expiration dates
Access control fields
Similar to corresponding fields in *data set* entry
Group fields
Names of groups authorized by propagation
Superior group for each authorized group
Group from which authorization derived for each authorized group
User fields (installation option)
Names of users authorized by propagation
Sources of user authorizations
Password fields
Similar to corresponding fields in *data set* entry
Account statistics fields (installation option)
Similar to corresponding fields in *group* entry

- The current definition includes more complex propagation of data privilege than in IMF. It is not clear to what extent this complexity is required so that views can be built upon views. It is apparently related to the fact that the content of access depends on the author of the granting view.

One can speculate that derivatives of System/R will become application subsystems like IMS. Such subsystems can be isolated as protected subsystems. A more likely accommodation is to employ the user and group information from the inventory together with message control programs and sign-on procedures common to the entire system. The data access mechanisms and tables would remain within the subsystem. It is not necessary that the data access rules of IMF and a relational subsystem be reconciled, but the extreme youth of the latter suggests considerable future evolution.

Network support

Remotely attached devices are treated in one of three ways by IMF: as sibling nodes, as interactive terminals, or as cluster controllers in tree-structure networks as in Systems Network Architecture (SNA).[18, 26] Although the IMF architecture was developed in parallel with and independently of SNA, it is intended to be compatible and generally is.

Remote computers with OS/VS control programs or the equivalent usually are regarded as sibling nodes in a computer network, and they are assumed to have IMF installed to protect their resources. Access to a sibling requires that the principal on whose behalf access is requested be identified to the sibling, together with a password.

Remote batch terminals are treated much like interactive terminals in the inventory data areas. However, the external interfaces for access to remote batch terminals are incompletely defined.

Intelligent terminals, which may include clusters of devices, can be treated much like multiterminal software subsystems. A user signing on to a device in a cluster in order to access the host system must identify himself to the host and be bound, via the subsystem support, to a set of messages originating at his terminal. There is a dependency that the cluster controller programs do not falsify the identity of the user or the originating terminal.

loosely coupled network support

Sets of users might have access to a specific computer in a network only via another network member. They must be able to define and control their access privileges over the network also. Envisioned is a network of interacting computer installations in

which each installation might be separately managed. Communications in the network are maintained by each processor's transmitting into the input streams of other processors.

Since all communication between nodes is via normal input streams, authorization can be checked at each target processor without extension of the IMF function already described. The only additions required to exploit remote processing are in the IMF command routines. Briefly, the additions permit a command to be issued at any node in the network for execution at another node, they support copying of inventory entries from one node to another, and they permit construction of an entry similar to an existing entry. The conditions under which network support is provided are:

- The part of the command processing pertaining to the current node will be executed as if the remote nodes were absent.
- For each other node invoked, a separate job or message will be constructed and transmitted. The user will be notified of its completion at the same node at which the command was issued.
- The command user must have adequate access privileges at each node for the services requested there.
- The output from batch jobs will be routed to the default output node in the current user entry at the node of execution.

This support assumes that the issuing user maintains the same USERID on all systems involved. This seems an acceptable restriction.

Table 8 *Contents of inventory subsystem entry*

Base portion
Subsystem name
Principal name for claims by subsystems
Author, creation and expiration dates
Audit trail name
Access control fields
Similar to corresponding fields in *group* entry
Protected transactions
Protection rules for transactions not named
Protected transaction names and flags
Statistics fields
For suggested items see page 111, Reference 28

terminal identification support

Terminal identification support was designed and implemented by Worley and others[27] subsequently to the description of the prototype, but it has not been described publicly. It is built around *terminal* and *relator* entries in the inventory. The terminal entry (Table 5) describes a device or a communications-line address. It includes descriptions of attached security hardware, an access control list, usage constraints and statistics, and whether or not the terminal is authorized for special-privilege users. The relator entry describes the uses that a specific user or group can make of a specific terminal.

Terminal identification can be introduced gradually into a functioning installation by progressively building up the necessary inventory entries and controlling the arming of checking features. Individual users remain unaware of terminal identification support except when they attempt to exceed their authorizations.

Terminal entries in the inventory can be created or destroyed only by users with the *administrative* or *operations* attributes. However, authority propagation is supported. Terminal entry can

be authorized to other users with the JOIN, CONTROL, USE, or NO authorities. An installation can indirectly control and monitor terminals that do not have hardware identification features by controlling and monitoring the use of communications lines. If automatic address answer-back features are installed, the installation can track and control specific terminals and:

- count terminal usage;
- track the date and time when the terminal was last used;
- limit or revoke access to a terminal by USERID, by group name, or according to the hour of the day;
- set a date when terminal access is to be revoked;
- limit sign-on violations for a terminal;
- build an audit trail of terminal usage;
- detect and document violations;
- identify terminals that are allowed system access.

intelligent terminal support

Currently IMF includes only very limited support for satellite processors. Of course, message control programs in the host can use the IMF assembly-language interfaces, and programs in the terminals can include passwords in their communications with the host message control program, but this solution is satisfactory only for cluster controllers in which the programs are under tight administrative control.

Table 9 Contents of inventory *transaction* entry

Base portion
Transaction or command name, subsystem name
Principal name for claims by this transaction
Author, creation and expiration dates
Name of first module to which linked
Priority
Audit trail name and flags
Access control fields
Universal access flags
Audit trail flags
Security code for output
Transaction password (optional)
List of authorized principals
Transaction statistics fields
Similar to corresponding fields in *subsystem* entry

The problem can be illustrated by two examples: one that can easily be brought under satisfactory control, and one that has not yet been adequately addressed. The first example is IBM's Subsystem Support Services (SSS)[28] as used for the 3650 Retail Store Subsystem.[29] Communication is initiated by a process on the host, so the principal involved is known at the host, the message control program is part of a specific application program, and only a well defined subset of the system data base can be accessed.

The second example is provided by a multisatellite, multiuser system, the Event Driven Executive.[30] The host communication facility for the Event Driven Executive is a message control program that provides the batch services of a central host to several satellites, each of which may have several concurrent users. A satellite application program can submit jobs to the host processor and can open and transmit user data sets to and from the satellite. Each command from the satellite can be ticketed with a (scrambled) identification of the user on whose behalf the command is being executed. That user will have logged on to a satellite terminal in the same manner as for a TSO log-on. The message control program can use these tickets to separate users.[31] If one user is not protected from another in the satellite, as in the Event Driven Executive, a security exposure exists. However, the risk is limited to other users of the satellite, who may invade each

other's resources on the host system, but not those of other communities. In some environments this exposure may be regarded as acceptable; in others, not.

Table 10 Contents of ACEB

User identification and current connect group
Current terminal and its security level
Current principal identification
Current account number
Current administrative privilege level
Current security level
Privileges for current session
Default universal access to be granted to new data sets
Minimal audit trail recording
Default security code for new data sets
Last program library member loaded

Installation management support

In any large computer installation, there are numerous administrative functions that are peripheral to any single user session but that support large numbers of users. IMF can significantly reduce the clerical effort related to some of these functions and simultaneously improve their integrity and management's control of them.

Addressed in the next two sections are the relative responsibilities and authorities of installation personnel and individual users in specifying the administrative interface for each user. There are several functions: the most important is control of account numbers; second in importance are descriptors, such as user names and mailing codes, on whose accuracy the smooth functioning of the installation depends; third are descriptions of how the user wishes the installation to process his work, such as terminal profiles and network routing codes; finally there are descriptive data of an informative nature, such as department names, which are not used by the system except in query utilities.

account control support

For each distinct account there is one inventory account entry, which contains the names of all groups authorized either directly or by propagation. As an installation option, the entry may also contain the name of each authorized user. The name of each included principal is accompanied by its author or superior group and the name of the host group from which the authority was propagated. Access to an account is limited similarly to access to a data set, except that only a user with the *administrative* privilege can authorize a group whose superior group is not authorized.

In addition, each group and each connect entry contains a list of the authorized accounts, enabling nonadministrative users to propagate the use of an account to subgroups and users, even though creation of accounts is privileged. In addition, account checking by the system during session initiation is expedited.

To create a new account and authorize a group to use it, the administrator must issue the command ADD ACCT (ACCOUNT-NAME) GROUP (GROUP-NAME). To authorize a group to use an existing account, the action is identical. Separately, an owner of a group must use the command ALTR GROUP to include the account in the group entry. Thus the administrative user cannot change the group entry, and the group owner cannot directly change the account entry. Their respective actions can occur in either order.

As an installation option, if a group is authorized to use an account, any user with at least CONTROL authority for that group can propagate the use of the account to connected users or subgroups.

user profile support

In the IMF prototype,[2] a serious error was made in the method of delegating responsibility for keeping accurate records about the identity and privileges of individuals. The accounting function and a large number of less critical administrative functions were negatively affected. It is difficult for a project manager to ensure that the system account numbers his subordinates are using are in fact valid. Nor may he become aware of an inaccurate account number until a large number of incorrect journal entries have been created. As an example of a less critical administrative function, note that it is extremely helpful to maintain the user name in the system in a standard format, as in telephone directories, but that it is probably futile to expect individual project administrators to adhere to such a standard.

The technical problem is how to centralize some administrative functions without giving up the merits of decentralized joining and authorization of users. This can be accomplished as follows: Each item in a user description is classified as to whether it may be altered by the user himself, by the owner of the user entry (the user who created the entry), or by any user with the *administrative* attribute. It is appropriate that some items can be set by the user himself and that they can be reset by an *administrative* user, who also can prevent further change except by an *administrative* user. With the items so labeled, the IMF commands for manipulating the user entry in the inventory can enforce the limitations.

A distinct but related problem is proliferation of the user lists required to administer an installation. It is not unusual to find five or ten distinct lists of users kept in data sets, with a larger number of shorter lists maintained clerically. Most users are named on several lists. A few of the lists are required by the operating system, and some are kept because others do not contain all required information. None of these lists are likely to be as accurate or up-to-date as is desirable because the cumulative administrative burden they impose is too great. All of the data can be kept in the inventory. If a separate list is required for a subsystem, its preparation from such a master list is trivial.

All the user profile fields are contained in inventory user entries. In some existing subsystems, such as TSO, one of a set of user profiles can be chosen when a terminal sesson begins; this capability could be continued by putting the profile data in connect entries, but very few users either want to maintain or are able to remember different profile data for different occasions. Several score profile items can be defined. They are classified in Table 4,

together with indications of the necessary authority to set items in each class. This type of central data structure, together with the implied constraints in manipulating commands, substantially solves the problems identified.

operations management support

IMF provides for improved control of the use of utility programs by limiting certain functions to users with special privileges. Most of the system utility programs for copying, modifying, and deleting data sets can run under the same access rule restrictions as any other program. However, utilities for compressing, copying, and initializing entire storage volumes must be restricted to *operations* users. (A means for recognizing such utilities is necessary.)

The inventory data provides an opportunity for easier administration in some areas. There should be a utility to dump, for any desired user, the user description, profile, and statistics—all the fields that define the administrative interface to the user. The utility would support such tasks as preparation of mailing lists for newsletters, and it would be restricted to users with the *administrative* or *operations* attributes. There might be a cross-reference utility program to display all occurrences of a name in the inventory. In each case, the name and the type of entry in which the name occurs would be indicated, together with the type of field in which the name is found. Such a program should be restricted to security officers, as should any utility that lists the entire contents of the inventory.

program library modification control

Program libraries, particularly those containing the modules of the operating system, present a special problem of administrative control. Replacements or temporary patches to introduce new functions or overcome malfunctions occasionally introduce new sources of malfunction. It is often desirable to track changes as they are made and save unmodified copies of programs immediately prior to a change.

In OS/VS, programs are stored as members of partitioned data sets. In IMF, as previously defined,[2] the smallest data entity controlled was a data set. Indirect control of records and fields by control of transactions, as described above, is not applicable. Instead, IMF can assist in achieving a desirable level of control by providing access specifications to modified utility programs.

Three classes of data are necessary for change tracking. Specification of the required treatment of a library member is properly part of the inventory. Copies of unaltered members and the alteration input command stream belong in a special change-tracking data set. Records of the author and the circumstances of each change can be directed to the system security audit trail, to the change-tracking data set, or both.

The change-tracking requests for a library can be stored in the inventory data-set entry. They can be specified by a user with access-control-list authority by means of a TRACK command. TRACK provides for specification of the data to be saved, where it is to be saved, the types of changes for which tracking is desired, and the retention period for copies created. The treatment can be defined separately for each member of the library; if a member is not named in the change tracking fields, a default action defined for the entire library applies.

Individual members of a partitioned data set are, for the most part, manipulated by system utilities and by the system linkage editor. Generally there are only a few such programs, suggesting an implementation with minimal system change. Only a principal with the *special* privilege can be authorized to change the protected library. Each of the utility programs must have an internal claim to that principal identifier and must include instructions to write the required change-tracking output. Although such an implementation is not particularly elegant, it is readily feasible.

Assessment of security

Security in an operating system requires that it permit precisely and exclusively the functions specified for it and also that the specification precisely reflect institutional needs. This paper has been concerned with the latter aspect, particularly with the function visible to an end user. Of course the protection mechanism provided by IMF can be subverted by a flaw elsewhere. A summary of the implicit assumptions follows:

- System integrity is assumed. Specifically, it is assumed that all control blocks in main storage have adequate protection, and that the use of system interfaces is restricted to properly authorized processes.[9, 17, 21, 32]
- It is further assumed that any installation-defined program segments installed for execution in the privileged state conform to the same rules as do operating system modules.[21] In particular, correct use of IMF interfaces is required.
- Central facility hardware integrity is assumed. Specifically, it is assumed that the hardware performs as specified, and that opportunities for misuse by operations or service personnel are controlled.
- Input data streams are presumed to contain no misrepresentation of terminal passwords or the identity of the accessor, and it is assumed that there is no improper substitution of accessors between occasions when passwords are checked.
- The author knows of no technical measures to control the actions of a small nucleus of system support programmers, so it

is assumed that appropriate administrative controls are in effect. Repeated but unscheduled involvement of an external auditor is recommended.

One attraction of security kernels is the hope that an operating system might be designed to be certifiably secure, in that certain well defined undesirable acts are provably impossible. Unfortunately, no such kernel has yet been defined for a system in which extensive data and procedure sharing is a key objective. The only alternative is exhaustive examination of penetration opportunities,[33,34] which at best is marginally satisfactory because it will inevitably be incomplete, and because it encourages the tendency to festoon the system with *ad hoc* contrivances.

The reader will have to make his own judgment. In the more detailed report[13] from which this paper is derived, the function of IMF is compared to a checklist accumulated from numerous requirements analyses.[9,10,17,35–39] In the area that IMF addresses, the author believes it to be quite effective, as well as easy to implement, because its authority-checking locations are restricted to a few subsystem alterations, and user identification is restricted to message control programs. In practice, it significantly enhances security by greatly reducing the number of persons technically qualified to invade the system.

Although it was not an explicit objective, experience with the IMF prototype has shown that it improves overall data integrity in an installation by automatically blocking many sources of human error.

Aid to installation management

The indicated objectives are substantially met. Comment here is limited to a few items not yet adequately addressed.

To the extent possible, the authority and responsibility to enter inventory contents is assigned to the user with the most interest in the accuracy and integrity of the particular data in question. An expected consequence is considerably higher overall accuracy in the administrative interface, with lower attendant costs and fewer frustrations for administrative personnel and for end users.

With current procedures, the responsibility of an installation security officer commonly degenerates into a clerical job. With IMF, he will act primarily as an auditor. In addition to the obvious management advantages, it is pertinent that the latter job responsibilities are much more satisfying than the former.

It is intended that an implementation would include a great deal of installation optionality. There can be a hierarchy of parameter

default values to minimize the data entry required of users. This hierarchy would include such optional features as terminal identification support, user profile items such as those normally required on batch job cards, and security features such as rules for the inclusion of audit trail records. The natural hierarchy would be:

- Many items, such as data-set universal access authority and maximal CPU time for batch jobs, would have default values that could be set by the user.
- Maximal authorities for many items could be set for each connected user by another user with JOIN authority for the group.
- If a default value or maximal authority were not specified explicitly, a value set by installation personnel would be used.
- If the installation did not supply a default value, an IMF-specified value would apply.

Throughout the hierarchy, the choices should be made with a tendency toward minimal authority for items that specify access to system resources, and maximal authority for items that are primarily for user convenience.

The network support for sibling nodes introduces no responsibilities at an originating node for checking authorizations at the target node. Each system in the network can install and administer its security mechanisms autonomously.

Features that can be exploited by an installation to improve administrative control and reduce clerical expenses include: checking all administrative parameters entered at the beginning of each session to reduce the frequency of invalid data in accounting logs, very simple procedures for specifying and revoking privileges, the instrumentation provided by the inventory for exploring usage patterns, and the improved potential for separating programming from administrative responsibilities in transaction oriented subsystems.

The most significant aid to installation management remains that previously identified[2]—the possibility of decentralizing the answerability for protection of the organization's resources to the individuals most suited to assume the particular responsibilities in question.

Performance

The opportunity to compare similar systems with and without IMF access control has not presented itself and would be a difficult experiment to mount on a meaningful scale. Moreover, the system drain, which consists primarily of inventory access, depends

on parts of the architecture not implemented in the prototype. So only conjecture is possible, augmented by the following comments:

- No extra input or output is necessary for transaction access control, and the traffic for all other entries, except those describing data sets and private storage volumes, is very low.
- For the latter entries, the most serious problem is serialization for updating when each data set is closed. Part of this problem can be avoided if usage statistics are not desired.
- If data-set inventory entries are integrated with the system directories, it might be possible to avoid any extra input or output for access checking. Alternatively, with the type of structure adopted for the prototype,[2] the input and output traffic can be reduced by paging the inventory, since the naming conventions induce good locality of reference.
- There is great optionality for both an installation and the user as to which IMF function is installed and invoked. Costs in channel programs, processing time, and data space are proportional to usage.
- Because of the hierarchical group structure, access lists tend to remain short, facilitating authority checks.

Experience with the prototype suggests that overall IMF consumption of system resources is less than one percent. An unsolved problem is that serialization of inventory accesses can significantly slow the response time for interactive users.

Conclusions

The administrative interface to a computer facility, which includes control of access to each valuable resource, must provide a large variety of functions to many classes of users, and it must be adaptable to a large variety of installations. This problem can be regarded as nothing more than a design challenge for a special purpose data base/data communications application. This paper, together with its predecessor,[2] demonstrates the feasibility of a consistent and economical implementation.

The extensions sketched here are straightforward. In fact, those extensions introduced as security features are entirely obvious in the light of analyses of requirements performed by other investigators. Considerable experience has been gained with several prototypes, which generally have been received favorably. Accordingly, a high level of confidence in the present proposal is justified.

However, the attempt to develop practical abstract definitions of the pertinent entities and their relationships is not yet as success-

ful as is either desirable or probably feasible. Consequently, the current architectural specification includes too many elements that are not forced by unifying concepts and are therefore somewhat arbitrary, particularly in some of the rules for delegation. Nor have the implementation specifications[13] been freed of all *ad hoc* contrivances. In addition, any OS/VS version must be made compatible with all other portions of that system, which are themselves evolving rapidly.

It is sometimes assumed that improved security is inevitably associated with extra inconvenience, an assumption that can become a self-fulfilling prophecy. IMF introduces extremely few new elements to the end user's environment until he attempts an unauthorized access. In fact, the inventory data structure creates an opportunity for significant improvement to the administrative interface between the installation and each user.

Sanguinity about security technology is always unwise and often dangerous. In a lecture some years ago, R. H. Courtney pointed out that the most common misuses of data are by individuals authorized to use the data for legitimate ends. IMF cannot address this type of exposure. Nevertheless, IMF does provide significant protection at remarkably low cost. The author is unaware of any alternative, in the area addressed, to the approach described.

ACKNOWLEDGMENTS

To E. Worley and K. Eckhardt for many conversations regarding installed versions of IMF, and in particular regarding accounting administration, terminal support, and volume support. Many others contributed to the prototype implementations.

Preparation of the manuscript was a demanding task; I am grateful for the help of S. Dwan, M. Price, and L. Ferguson.

CITED REFERENCES

1. N. R. Nielsen, "Computers, security, and the audit function," *AFIPS Conference Proceedings* **44,** 947–954 (1975).
2. H. M. Gladney, E. L. Worley, and J. J. Myers, "An access control mechanism for computing resources," *IBM Systems Journal* **14,** No. 3, 212–228 (1975).
3. *OS/VS2 MVS Resource Access Control Facility (RACF) General Information Manual*, IBM Systems Library, order number GC28-0722, IBM Corporation, Publications Development, Department D58, P.O. Box 390, Poughkeepsie, New York 12602 (1977).
4. For Project MAC background information, see the following references: L. L. Selwyn, *BUYTIM: A System for CTSS Resource Control*, Project MAC-M-379R (July 1969); D. H. Vanderbilt, *Controlled Information Sharing in a Computer Utility*, Project MAC TR-67, NTIS accession number AD-699 503, National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161 (1969); R. C. Owens, Jr., *Primary Access Control in Large-Scale Time-Shared Decision Systems*, Project MAC TR-89, NTIS accession number AD-728 036, National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161 (1971).

5. A. M. Lieberman, "An approach to data security through data set access control," in *Data Security and Data Processing* **4**, *Study Results: Massachusetts Institute of Technology*, IBM Systems Library, order number G320-1374, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).
6. M. V. Wilkes, "On preserving the integrity of data bases," *Computer Journal* **15,** No. 3, 191–194 (1972).
7. A. B. Wilson, Jr., "A disk data set security system," *Proceedings of SHARE XLIV* (Los Angeles) **1,** 437–444 (March 1975).
8. B. Schrager, "Access control and security system," *Proceedings of SHARE XLIV* (Los Angeles) **1,** 445–451 (March 1975).
9. J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE* **63,** No. 9, 1278–1308 (September 1975).
10. M. V. Wilkes, *Time-Sharing Computer Systems*, American Elsevier Inc., New York (1975).
11. H. M. Gladney, *An Access Control Mechanism for Computing Resources—II Commodity Resources*, Research Report RJ 1825, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (1976).
12. J. J. Myers and J. P. Considine, "MVS Automatic Storage Hierarchy," *Proceedings of SHARE XLVI* (San Francisco), 147–164 (February 1976). See also J. P. Considine and J. J. Myers, "MARC: MVS archival storage and recovery program," *IBM Systems Journal* **16,** No. 4, 378–397 (1977), and *OS/VS2 MVS Hierarchical Storage Manager: General Information*, order number GH35-0007, IBM Corporation, Information Planning and Editing, Department 21E, P.O. Box 1900, Boulder, Colorado 80302 (1977).
13. H. M. Gladney, *Administrative Control of Computing Service*, Research Report RJ 1852, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (1976).
14. P. G. Neuman, L. Robinson, K. N. Levitt, and A. R. Saxena, "A formal methodology for the design of operating system software," Chapter 3 in *Current Trends in Programming Methodology* **1,** R. T. Yeh (Ed.), Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1977).
15. G. J. Popek and C. S. Kline, "Verifiable secure operating system software," *AFIPS Conference Proceedings* **43,** 145–151 (1974).
16. B. W. Lampson, "Protection," *Proceedings of the Fifth Annual Princeton Conference on Information Sciences and Systems*, 437–443 (March 1971).
17. F. M. Stepczyk, "Requirements for secure operating systems," in *Data Security and Data Processing* **5,** *Study Results: TRW Systems, Inc.*, IBM Systems Library, order number G320-1375, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).
18. H. R. Albrecht and K. D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture Perspective," *IBM Systems Journal* **15,** No. 1, 53–80 (1976).
19. *CICS/VS General Information Manual*, IBM Systems Library, order number GC33-0052, IBM UK Laboratories Ltd., Publications Department, Hursley Park, Winchester, Hampshire SO21 2JN, England.
20. W. S. McPhee, "Operating system integrity in OS/VS2," *IBM Systems Journal* **13,** No. 3, 230–252 (1974).
21. *IMS/VS Version 1 General Information Manual*, IBM Systems Library, order number GH20-1260, IBM Corporation, P.O. Box 50020, Programming Publications, San Jose, California 95150 (1977).
22. D. K. Hsiao and R. I. Baum, "Information secure systems," *Advances in Computers* **14,** 231–272, Academic Press, Inc., New York (1976).
23. E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM* **13,** No. 6, 377–387 (June 1970).
24. M. M. Astrahan et al., "System R: a relational approach to data base management," *ACM Transactions on Data Base Management* **1,** No. 2, 97–137 (June 1976).

25. P. P. Griffiths and B. W. Wade, *An Authorization Mechanism for a Relational Data Base System*, Research Report RJ 1721, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (1976).
26. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4–23 (1976).
27. E. L. Worley, K. L. Eckhardt, R. Y. Shimizu, and O. E. Palmer, Jr., private communication.
28. *IBM System/370 Subsystem Support Services User's Guide*, IBM Systems Library, order number GC30-3022, IBM Corporation, Programming Publications, Department 63T, Kingston, New York 12401 (1977).
29. *IBM 3650 Retail Store System Introduction*, IBM Systems Library, order number GA27-3075, IBM System Communications Division, Publications Center, Department E01, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (1977).
30. D. L. Raimondi, H. M. Gladney, G. Hochweller, R. W. Martin, and L. L. Spencer, "LABS/7—a distributed real-time operating system," *IBM Systems Journal* **15,** No. 1, 81–101 (1976). This system is now called the Event Driven Executive.
31. R. W. Martin and E. Worley, private communication.
32. L. Smith, *Architectures for Secure Computing Systems*, NTIS accession number AD-A009 221, National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161 (1975).
33. R. R. Linde, "Operating system penetration," *AFIPS Conference Proceedings* **44,** 361–368 (1975).
34. C. R. Attanasio, P. W. Markstein, and R. J. Phillips, "Penetrating an operating system: a study of VM/370 integrity," *IBM Systems Journal* **15,** No. 1, 102–116 (1976).
35. "Summary of findings: Massachusetts Institute of Technology," in *Data Security and Data Processing* **2,** *Study Summary*, IBM Systems Library, order number G320-1371, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).
36. "Elements and economics of information privacy and security," in *Data Security and Data Processing* **3,** Part 2, *Study Results, State of Illinois*, IBM Systems Library, order number G320-1373, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).
37. "Recommended security practices," in *Data Security and Data Processing* **3,** Part 2, *Study Results, State of Illinois*, IBM Systems Library, order number G320-1373, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).
38. R. H. Scott, "Requirements for secure operating system features—MIT as an example," in *Data Security and Data Processing* **4,** *Study Results: Massachusetts Institute of Technology*, IBM Systems Library, order number G320-1374, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).
39. R. C. Daley and J. P. Donohue, "Security and authorization—semantics and examples," in *Data Security and Data Processing* **4,** *Study Results: Massachusetts Institute of Technology*, IBM Systems Library, order number G320-1374, IBM Corporation, Technical Publications—Systems, Department 824, 1133 Westchester Avenue, White Plains, New York 10604 (1974).

Reprint Order No. G321-5069.