The Source of Authority for Commercial Access Control

Jonathan D. Moffett and Morris S. Sloman
Imperial College of Science and Technology, London

Access control systems for a company's computers should mirror the organization's internal control systems, based on the delegation of authority.

Models of access control have historically been based on "friendly" academic or "autocratic" military requirements. In neither case has the question of the source of authority for access control decisions arisen. Commercial security policies differ from both of these in important respects, being based on control principles derived from auditing practice and legislation.

Authority for the decisions which permit users to access resources needs to be considered explicitly and reflected in the policy model. Access control policies define the rules which regulate how people (and programs acting on their behalf) can access resources in a computer system. Five components make up an access control policy:

(1) *Users* may be people sitting at a terminal or workstation or the processes which run on their behalf within the computer system. The policy identifies what a user is allowed to do.

(2) *Resources* are the programs, services, or data accessed by users. The policy specifies what resources a user can access.

(3) *Operations* are the specific operations which can be performed on a resource. Each resource type has a set of operations it can support. It is insufficient for the access control policy to specify that a user can access a resource; it must specify which operations the user is permitted to invoke on the resource. For example, one user may be permitted to update a file, whereas another user may only read the file.

(4) *Authority* is the legitimate power to make policy decisions. This article is mainly concerned with describing the way in which authority is delegated so as to ensure that access control is implemented in accordance with the policies of the management of an organization.

(5) *Domain* is the boundary, containing resources or users, within which the authority applies. For example, a manager typically has authority over the resources and people in a department.

We distinguish three levels of policies in this article, although we make no rigid distinction among them:

- *General policies*, which do not relate to a specific organization, such as the decision to give access control administration to security administrators.
- *Specific policies*, which are high-level decisions in a particular organization, such as the decision to give access control administration of the marketing department of XYZ Co. to the security administrator in the data processing department.
- *Access rules*, which are the specific decisions, made at a lower level, to allow particular users access to particular resources. The term is equally applicable to capabilities or access control lists.

User identification. A critical issue relating to access control is identifying users. Computers do not naturally recognize people, and it was necessary to invent methods of enabling them to do so. The identification of people involves three stages:

- Telling the computer who is using it: identification.

- Proving one's identity: authentication.
- Associating a person with a process.

Identification is achieved by creating a register of users and assigning each person a user ID.

An authentication method, such as a password, magnetic card, or signature verification system, helps ensure that the user is in fact who he or she claims to be. It may be backed up by supporting evidence, such as the identity of the terminal from which the user is logging on. Authentication can be used at entry to the system (log on) or at other times during a user's session.

Since all actions of a computer are carried out by means of processes, the only way to associate a particular user with a computer's actions is by binding his or her identity to a process. So, some (but not all) processes in a computer are associated with a person. For example, many file accesses are carried out in response to a request by a person, and the process carrying out the request takes on that person's user ID for the duration of the request.

In a centralized computer system, it is assumed that once a person's identity is associated with a process, all access control decisions could be related to the person's identity. However, in a distributed system the identity of a process is a real issue. There may not be a single controlling operating system which can vouch for the identity of each process. Therefore, it was necessary to devise some means of authenticating processes to each other. The main contributors to this, Needham and Schroeder,[1] introduced a mutual authentication protocol for independent processes using a trusted authentication server as an intermediary. These techniques for the mutual authentication of processes have become a recognized part of the access control mechanisms for distributed systems.

Authority. The access control policy specifies the guidelines or general rules for deciding what operations—users in a domain—can perform on resources. In addition, the policy should specify the source of authority, that is, how access rights can be given to users. Authority relates to the management of both users (people) and resources.

This article is concerned with how access rights are allocated to users in an organization. In general, an access right moves through an organization by one person giving it to another. In a commercial organization it is important to consider two things: Does the giver have authority over the resources to which he is giving access? Does the giver have authority to give access to the recipient? Only if both conditions are met can the access right be given.

Surveys indicate the actual threat from computer fraud is relatively small despite the potential for abuse.

We will explain why authority is important for commercial systems and will show how it can be specified and checked.

Commercial security requirements

The three main influences which motivate the need for security in commercial organizations are protection of assets, auditing practice relating to public ownership of a company, and legislation.

Protection of assets. The organization's assets, whether money, goods, or proprietary knowledge, need protection. The increased reliance upon computers to support operations or even carry them out implies that, to an increasing extent, protection of assets implies protection of computer resources against loss or corruption.

The threats to an organization's assets through its computers are by now well-known. They include

- physical damage to computer equipment;
- loss of vital operating data as a result either of physical damage or user action, whether accidental or deliberate;
- unauthorized disclosure of confidential information, directly or by inference; and
- fraud.

Fraud in particular has attracted considerable attention. The potential to manipulate systems from a distance to steal from a company excites the public imagination. However, its importance should not be exaggerated. Surveys of actual computer-related fraud incidents[2] indicate that the threat of computer fraud, even after allowing for under-reporting, is relatively small. Its actual overall incidence is a tiny proportion of the frauds committed without the aid of specific computer techniques. But the potential for fraud in computers, if they are not well protected, increases with the ease with which they enable manipulation of assets.

Auditing practice. Commercial organizations exist mainly to make profits for their owners. In some cases, such as privately owned companies and professional partnerships, the owners are also the managers; but in the case of public limited companies the owners are shareholders who do not take part in the running of the company.

The possibility of shareholders losing because of mismanagement of the company has long been recognized. Therefore, all developed countries have laws requiring regular audits of each company to verify that the company's profits and assets are truly stated in management's reports to its shareholders.

Auditors can and do verify the assets of the company directly and vouch for individual transactions (a *balance sheet audit*), but auditing practice has increasingly emphasized the auditing of the computer system itself (a *systems audit*). In this approach, the auditor assesses the validity of the operations which manipulate the resources in the computer systems. This means that all transactions which can be invoked by users are validated to ensure that they manipulate the resources in a constrained way so as to maintain the integrity of the resources. If it can be shown that the system can only perform valid transactions, then it is not necessary to log and check individual transactions to the same extent.

Legislation. The Foreign Corrupt Practices Act was passed by the United States Congress in 1977 in response to the worldwide scandal caused by the Lockheed bribery case. The company had evidently obtained much of its overseas business by corrupt means. The act applies to all foreign subsidiaries of US-owned multinational companies. Mostly it concerns outlawing bribery and other corrupt practices, but it also contains a requirement that all of these subsidiaries must have effective systems of internal control over their transactions and assets. This includes the full recording of all transactions. Because of the severe penalties meted out in the past to firms which have committed ethical offenses in the USA, the Act has been taken very seriously by auditors, both internal and external, of US-based multinationals.

The data protection laws which now exist in most Western countries (such as the UK Data Protection Act, 1984) have also provided a strong motivation for security, as they require access to personal data to be allowed only when specifically authorized.

Commercial security controls

Access control mechanisms. These mechanisms permit access by a user to only those resources for which he or she has received authorization. On older systems, this may be done by associating passwords with individual resources, but the method is unsatisfactory because of the difficulty of sharing and lack of personal accountability.

Mechanisms associated with individual user identities provide potentially better control. Access control lists hold the rights in association with the resources to be accessed. Capabilities are rights held by the user until he or she comes to exercise it.

However, our main concern is not with access control mechanisms and the proof of their security, but rather with access control policies and how access is authorized.

Internal organization controls. Internal controls are not specific to computer systems, nor are their principles laid down in legislation. Nonetheless, the main principles are common ground among auditors and accountants.[3] The principles which most concern this discussion relate to transactions affecting assets:

- The transactions must be authorized, meaning they must be carried out by someone to whom authority has been duly delegated.
- As far as practical, there must be segregation of duties and responsibilities so that no single person is in a position to misappropriate assets. A typical example of segregation of duties in a computer system is the requirement that creation of a batch of transactions on a computer be carried out by a different person from the one who approves and initiates them.

Segregation of duties in a computer system can be difficult if the manager administers access rules for his section.

Segregation of duties can be difficult to achieve in a computer system if the manager of a section is also responsible for administering access rules for the data used by his section. For example, if an accounting section consists of a supervisor and several clerks, typically the clerks will have authority to input transactions but not to approve them, while the supervisor can approve transactions but not input them in the first place. If the supervisor also administers access rules, segregation of duties cannot be enforced because the supervisor could change the rules to obtain the power to input transactions. The best way of avoiding this is to designate a security administrator with no line authority, but with the authority to administer data access rules on behalf of line management. The security administrator does not require, and should not have, access to the data.

Other approaches—why they are not adequate. In contrast with the commercial world, the worlds in which access control have been developed take a simple view of authority. Initial formal approaches to access control were developed by the academic world, but military requirements have more recently dominated the direction of development. A 1981 survey of formal models of computer security[4] concentrated on provable security. Most recent papers have confirmed this trend.

At first, models of access control reflected the academic's friendly view of the world, in which freedom of access to information was a cornerstone. Access controls were minimal,[5] concerned more with controlling processes than people. Where people were included,[6] the examples used were from a world in which the typical security threat was a student wanting to take an advance look at an exam paper or alter grades. This world assumed widely scattered information ownership, and that information should be available unless there is a positive reason to protect it.

The next type of model has dominated the past few years, since the military grasped the fact that physical security alone would not suffice to protect their computer systems. The *USA Department of Defense Trusted Computer Evaluation Criteria*,[7] first published in 1982, has served as its beacon. In this world of rigid compartments, decisions about access to information depend primarily on its security classification, enforced by mandatory security mechanisms. Flexibility is deliberately almost entirely designed out of the system. For the typical person handling classified data under military security conditions, security decisions are handed down by a superior officer; they are obeyed, not made.

Military security emphasizes preventing access by unauthorized people, whereas commercial security emphasizes making sure people authorized for some access do not invoke unauthorized transactions. A high proportion of recent research into access control has focused on implementations of provably secure software to meet the DoD model.

So the two paradigms of access control, although poles apart in attitude, have had one quality in common: Neither has concerned itself much with the source of authority for access control decisions, in one case because access control is treated as a necessary evil to be minimized, and in the other case because decisions are made from above and are not to be questioned. This attitude has probably been encouraged by the very definition of "policy" to be found at the front of most works dealing with security.[8] "Policy" is described as a set of rules for providing or maintaining security, but the source of these rules is not referred to.

Many people have assumed that security policies for commercial systems are either less friendly versions of academic policies or military policies with fewer teeth. Neither is true. Considerations for commercial security policies differ in quality, because the principles of internal control take a much more sophisticated view of authority. Effective internal control can only be maintained at a reasonable cost if the access control system allows for the controlled devolution of authority to designated people. This is a more stringent and more complicated approach than had been used until recently.

A recent article[9] has recognized the problem and defined a set of rules for commercial security in a system which, if implemented, would maintain internal control by implementing principles of verification of integrity, authorization of procedures, and segregation of duties. But it has not considered the way in which authority is distributed in the system.

Authority

We can define authority as the legitimate power to control or manage. Two types of authority affect access control: authority relating to people and over resources. A person (usually a security administrator) who issues a user access rights to perform operations on a resource needs both authority to grant access rights to that user and authority over the resource. He or she has two domains of authority:

- The *organizational domain* defines those users in an organization, normally defined by their positions (see below), for whom the person with authority can perform security administration services. This may be unlimited—the whole organization—or it may be limited to a particular section of the organization, such as the marketing department. The security administrator can only give access rights to a person occupying a position within his or her organizational domain.
- The *resource domain* defines the set of operations on resources to which the security administrator has authority to grant access rights.

The system must permit a security administrator to give an access right if and only if the recipient occupies a position in the administrator's organizational domain and the resource falls in the administrator's resource domain.

The source of authority. Traditionally, the source of authority over use of resources or assets is vested in their owner. For example, many file systems consider the creator of the file the owner, who can issue access rights to other users and delete the file. However, in a commercial system it is meaningless to consider the creator of a bank account file the owner of that file, with authority over it.

Segregation of duties applied to computer systems defines the organizational domain of the security administrator to exclude his or her own position.

Strictly speaking, the shareholders of a company own the resources in that company. In many commercial organizations, ownership is separated from management, so the owners (shareholders represented by a board of directors) delegate authority to management. It is useful to distinguish between full control and limited control such as a security administrator's control over access rights to a file. We will use the term *owner* to represent the person (or position) who has full control over a resource. You should recognize that this distinction begs some questions (such as, what is "full"?), but it stands up reasonably well in practice.

Most organizations are structured hierarchically, with a board of directors that delegates authority to top management, who in turn delegate some authority to departmental management, and so forth. The eventual owner of the resource is thus the person to whom full control of it has been delegated.

The process shows most clearly in the delegation of monetary budgets. The board of a company agrees on a total budget for the coming year. The money in the budget is then divided between major parts of the company, such as manufacturing, marketing, and research, in proportions determined by the board. Each major division divides its available budget appropriately among the activities it carries out. In most well-run companies, the budgeting process is carried out in a formal and visible manner, with each manager knowing the extent of his or her budget and the limits of his or her authority.

For obvious reasons, delegation of authority appears in its most developed form in the case of monetary budgets. However, the concept of delegated authority also applies in other areas, such as the management of personnel in the organization. It has also proven a useful concept for data processing.

Numerous problems arose in the earlier days of data processing, when the computer department effectively "owned" all of computing; often computer people gave users what they thought would be good for them, not what the users actually wanted. The idea therefore arose that applications and data should be treated in the same way as other company assets, with ownership being well-defined. Thus, the director of a marketing department has ownership of its applications and data, but can delegate that ownership. The director may delegate internally to subordinates, such as a user project manager. He or she may also delegate limited authority externally to the data processing department, such as the tasks of development of new systems and operation of production systems.

In well-developed companies, the delegation of authority (ownership or access) over computer resources is nearly as formally defined as for monetary budgets. The source of authority for use of these resources is a chain of formal decisions which can be traced back to decisions of the board of directors managing the company for its owners, the shareholders.

Three classes of rights can be delegated:

- *Ownership* gives total control over a resource.
- *Give-right* permits the holder to give an access right to a third party. A give-right applies to a specific operation on a particular resource instance. The owner of a resource may delegate the authority to issue access rights to that resource to a security administrator.
- *Access right* gives the authority to perform a specific operation on a particular resource instance. In general, the possession of an access right does not automatically permit the holder to transfer it to another user.

Possession of one type of operation right may imply another. For example, write access to a file may imply read access in some systems.

The authority of both the owner and the security administrator can be traced back to formal decisions of delegation of authority.

Access control systems sold commercially naturally recognize the facts of life, and mostly represent lines of authority reasonably satisfactorily. For example, all the major access control systems for IBM's large systems, such as IBM's own Resource Access Control Facility (RACF), define a security administrator role separate from normal data access or ownership. It is possible to set up the resource domain for the security administrator so that he or she cannot obtain access to data, but must rely upon the authority of a different security administrator for this purpose. Only the rarely used bootstrapping user ID, whose use can be monitored, gives unlimited power. Even the rather old Multics system[10] recognizes the possibility of separating access from administration, although it does not claim to do so in a very satisfactory fashion.

Formal academic models, on the other hand, have so far given little recognition to control principles as outlined above. Lampson[5] tightly bound the security administrator's authority to give-rights with the access rights, by means of a special property which, when associated with an access right, also allows that right to be passed on. Snyder[6] took the same approach, with a "grant" exercisable only if the grantor possessed the relevant access right. A more recent article[11] recognized that the security administrator's authority to give access should not be bound in with the access right itself. In this model, which dealt with distributed authority, the giving of access was independent of the access right and dependent upon the degree of trust between different authorities. However, the setting up and changing of the trust relationship was outside the scope of the model.

Who guards the guardians? This question did not arise with computer systems—it has been asked regularly since Plato's time. Its age indicates that we cannot expect to find a complete and final answer.

The question translates naturally to the subject of access control. What is the point of giving someone the authority to grant access rights to other people, but refusing access to that person? He or she can always change the rules to obtain access.

What prevents a larcenous bank clerk from taking the bank's money? The answer is, sometimes nothing. But the risk can be

- limited in magnitude,
- reduced in likelihood by making it more difficult, and
- made detectable, so that the culprit can be caught or, better, deterred from doing it in the first place.

The risk can be limited in magnitude by placing upper limits on a clerk's authority. Similarly, the resource domain for a security administrator does not need to include all the data in the system, but only that data whose access the administrator must control.

The risk of theft can be reduced by making it more difficult. Most organizations have an upper limit for the financial authority of one person and require the cooperation of two people for authorizing sums above the limit. This can also be implemented in computer systems—the segregation of duties principle again. A simpler method of doing this is to define the organizational domain of the security administrator's powers to exclude that position; someone else should control the administrator's access.

The risk can only be limited, not removed, because any security system must be bootstrapped in, allowing the bootstrapper the possibility of illicit access. However, the risk can be confined to well-defined situations monitored by personal supervision.

The risk can be made detectable. Daily reconciliations and regular audits in a bank are intended to ensure that, if a clerk does make off with a handful of notes, the loss will be rapidly detected and traced to its perpetrator. Similarly, the actions of a security administrator can be logged and should be monitored regularly by someone else. Even if the administrator can turn off the logging, it is normally possible to ensure that the act of turning off logging is recorded securely, such as on a hard-copy console log or WORM (write-once read-many) media. The bootstrapping process is the most vulnerable to suppression of logging and, therefore, needs the most supervision.

There is nothing new, then, in the question of who administers the security administrators. Although not solved, the problem is limited by narrowing the scope for misbehavior, recording all significant actions, and concentrating personal supervision on the main point of vulnerability in the process.

Policy

Policy making. Once we have recognized the need to incorporate the authority structure of an organization into any model of access control, the inclusion of the organization's policies becomes essential. Even if a manager has a monetary budget, he or she still does not have unrestrained freedom to spend the money. Policy decisions made by higher management constrain the manager to act in ways consonant with the interests of the company as a whole. The manager recognizes the force of these decisions because they have been made in accordance with the authority structure of the company. A policy decision made by the marketing director is not binding on the manufacturing division unless the board has given the director authority in that particular area. Similarly, access control policies are rules made by the appropriate authority which act as constraints on the giving of access rights.

It helps to be rather more specific about policy-making, while recognizing that this prescriptive picture may be false for many organizations. Most organizations have a steering committee or other body to which the board has delegated the task of overseeing and policy-making for the computing activity. Access control policies are those decisions of the steering committee which affect the way that day-to-day access control is carried out. They can thus be seen as grounded firmly in the authority structure of the organization, rather than arising spontaneously.

Examples of general access control policies include:

- Access rules should be made about positions, not people, in order to ensure that they continue to be valid after personnel changes.
- A security administrator should only be able to grant access to people within his or her defined organizational domain.
- A security administrator should only be able to grant access rights to resources in his or her defined resource domain.
- A manager should be able to specify the people under his or her control who form an organizational domain and delegate access control authority over this domain to a security administrator.
- The person who has access to a resource domain (such as a directory) has access to all resources and subdomains contained in that domain unless there is a more specific rule to the contrary.

Needless to say, a typical organization will not necessarily adopt all of the above policies.

Policies are either mandatory or discretionary. Security administrators and users are forced to follow mandatory policies, preferably because the policies are enforced by the system, but otherwise by means of normal organizational principles of internal control. Discretionary policies, on the other hand, leave the system administrator and possibly the user with some discretion in applying them.

An example of a mandatory policy is that only specified operations (transactions) can be invoked on authorized resources.[9] This is implicit in object-oriented systems, as resources and their operations are specified as a single entity. Otherwise, explicit security mechanisms are required.

Whether owners give out access rights or delegate this to security administrators is itself a security policy decision.

Role of security administrators. We discussed the role of security administrators briefly under authority, introducing the concepts of organizational and resource domains. This section describes how the concepts can be used to define security policy.

Whether owners give out access rights or delegate this to security administrators is itself a security policy decision. In both cases, the system should perform the following checks when an access right is given to a user:

• Does the giver administer the recipient? In other words, is the recipient in the giver's organizational domain?

• Does the giver have authority over the resource for the requested operation? In other words, is the resource in the giver's resource domain?

Interestingly, Snyder's formal access model[6] deals explicitly with administration rights over the user. His "grant" relationship between two users defines an organizational domain. However, an administrator's resource domain is defined as those resources to which the administrator has access rights. This is a practical policy, but not one we recommend.

A number of possible policies relate to organizational domains:

• In universal organizational domains, an access right to a resource in the giver's resource domain can be given to anyone.

• In flat organizational domains, individual users are placed into a single organizational domain administered by a security administrator.

• In hierarchical domains, users are structured by a manager into subdomains reflecting departmental structure.[12] Complete subdomains are delegated to a security administrator.

Corresponding policies can relate to universal, flat, and hierarchical resource domains.

Explicit authorization of domain policies and the domains of individual security administrators is a required part of effective internal control, while precise description of them is a prerequisite for a model of an access control system.

People and positions. In many organizations, access policy applies not to individuals but rather to the positions they occupy within the organization. When someone moves to a different position, most of that person's access rights should not travel with him or her, but be inherited by his or her successor. The reason for this is that policy is not normally expected to change with changes in personnel. There are, of course, many counterexamples to this, so the following general policy allows for exceptions:

• As far as possible, access rules should refer to positions, not people.

In practice, this policy has helped reduce routine work by minimizing the changes in rules needed when people change positions.

Ownership of resources. Internal control principles require that resource ownership be unique and derive from delegated authority down the management structure. Just as budgets can be passed down an organization by delegation of authority, so too can ownership of resources such as data and distributed hardware.

Access control policy needs to take into account how much freedom people should have in relation to the resources with which they work. For example, the possible policy statement

• A user may grant another user any kind of access right to data he or she has created.

might be appropriate for a project which is not at all sensitive, but totally inappropriate for someone using end-user computing tools for development of a secret product. The stronger statement

• A user may grant another user any kind of access rights and give-rights to data he or she has created.

will be acceptable only to liberal regimes.

Therefore, there may be limits to the extent to which ownership of resources such as data should be passed down an organization. Most organizations have managerial positions, usually defined by the possession of a budget. These may also be the level down to which resource ownership should be allowed. Liberal institutions may place no restrictions in some cases.

A simulation model

Need for formal modeling. The discussion of policies above makes it clear that their consequences can be quite complicated and, indeed, that the policies themselves can be difficult to formulate in normal English. So, there is a natural incentive to formalize the way in which they are described. There are two main reasons for creating a formal model of access control policies:

• The model provides a precise specification from which implementation can be carried out.

• The model can be used to validate security policy by ensuring that its consequences will actually be those intended.

We can use simulation to explore the consequences of possible access control policies. We can also use it to find security weaknesses in existing access control systems while avoiding the need to handle unmanageable quantities of data.

The approach used by Snyder[6] was to trace the possible flow of access permissions through the model access control system, in order to determine whether undesired consequences could follow. This was possible because his models were self-contained, with the giving of access bound to the access right itself.

Although it is a step forward to decouple the giving of access from the access right,[11] if we take the delegation of authority for security administration outside the formal model, we cannot ask questions about how rights can flow within the model. The model can no longer be used for simulation.

What we need is a formal model for specifying the security policies of an organization. The model must incorporate the five components identified earlier, namely, users, resources, operations on resources, domains, and the authority structure of the organization. As a step towards formal modeling, below we give an outline of how we used a Prolog program to represent comparatively complex policies. The model can be used to query both delegation of authority and ability to access a resource.

Indirect relations. We require some definitions to specify the hierarchical relationships *manage*, *administer*, and *contains* between people and resources.

A relation is

Transitive if: if *x rel y* and *y rel z*, then *x rel z*.

Reflexive if: *x rel x*

Consider:

- A marketing director *manages* a sales manager and the sales manager *manages* a salesman.
- A root file directory *contains* the marketing files directory and the marketing files directory *contains* the marketing files.

If *manage* is transitive, then the marketing director *manages* the salesman. However, it is a good principle that everyone should know who they work for, and in this case the salesman's immediate superior is the sales manager, not the marketing director. So, we may well decide (as a matter of policy) that *manage* should not be a transitive relation. Similarly, if *contains* were transitive, then the root file directory would *contain* the marketing files, which is not strictly correct.

It is useful for each hierarchical relation to have both indirect and direct versions, where the indirect version is transitive and the direct one is not. It would then be true that

- The marketing director *indirectly-manages* the salesman.
- The marketing director *indirectly-manages* the sales manager.
- The sales manager *indirectly-manages* the salesman.

But the only true statements for *manage* would be

- The marketing director *manages* the sales manager.
- The sales manager *manages* the salesman.

Simulation can be used to explore the consequences of possible access control policies.

We need to be able to reflect the policy:

- The marketing director delegates administration of everyone in his department (including himself) to the security administrator.

This can be arranged simply by ensuring that *indirectly-manages* is a reflexive relation:

- The marketing director *indirectly-manages* himself or herself.

Then

- The security administrator *administers* a position if the marketing director *indirectly-manages* that position.

If *indirectly-contains* is also defined as a reflexive and transitive version of *contains*, then we can express the following policy without having to worry about the level of the directory: A user has access to a data file if he has access to a directory which *indirectly-contains* the file.

Introduction to the simulation model. The examples below are extracted from a model written in Logic Programming Associates' Micro-Prolog[13] and run on an IBM PC. At this stage, we have only modeled an imaginary installation. Prolog is a very suitable language for modeling access control because it enables us to formulate statements which directly express the questions we wish to ask, such as

has-right(person resource access-right)

We found that the structure of the Prolog program fell naturally into three parts, reflecting the policy levels identified in the introduction.

- General policies about organization and resource authority which apply to all installations and define the interpretation of concepts such as *manage*, *administer*, and *contain*.
- Specific policies which define the relationship between the resources and people in the imaginary organization to which the model applies. These are relatively invariant in time.
- Access rules define the state of affairs at a particular point in time—what files, positions, and people are known to the system, and what access rights they have.

We made maximum use of parameterization and generalization in the model, gaining flexibility at the expense of readability. Our examples here have been amended to make them easier to read. Also, to enhance readability, we used the convention that predicates are in italics, while variables are in lowercase and constants that bind them are in uppercase, such as "position1" and "MARKETING DIRECTOR."

Note that all grants of access rights are reinterpreted and revalidated at the time of an access request. This clarifies the way in which an access decision is made and involves no loss of generality. It does, however, have implications for the performance of the model. A working large-scale simulation model would validate all grants of access on a once-only basis, updating the database as it did so.

Access rights. The overall aim of the model is to define who has access rights, given certain policies and facts. Possession of a right allows a specific operation on the resource. The corresponding predicate is *has-right*. Most queries on the model will be of the form

has-right(person data right) ?

and the main policy statements are of the form

has-right(person data right) if ......

Operations. This model, for simplicity of illustration, recognizes only one type of resource—data. There is a hierarchy of directories, with the operations being performed on data files. The possible operations are R, W, C & D, standing for Read, Write, Create & Delete. In this model, Write access does not imply Read access, although the implication could be easily included.

GIVE-R GIVE-W GIVE-C GIVE-D are the give-rights which correspond to R, W, C & D. The model has to be told specifically how to associate access rights and give-rights. This is done by direct facts included in it:

gives(GIVE-R R)
gives(GIVE-W W)

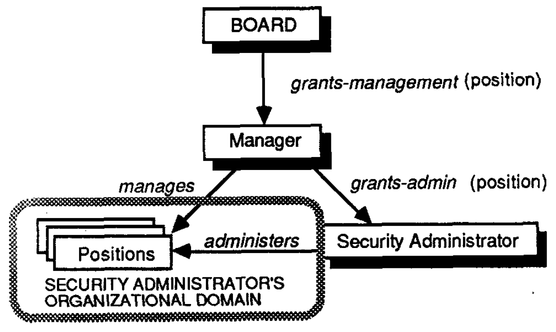

Figure 1. Organizational authority.

gives(GIVE-C C)
gives(GIVE-D D)

People and positions. There is a general policy that access rights refer to positions, not people, but it is people who grant accesses and perform operations. The model has to be told how to relate people to their positions. For simplicity this has been done by direct insertion of facts in the model, although it would have been more realistic to show that this, too, is an operation requiring authority:

occupies (person position)

Organizational authority. For the organizational domain the main concepts are *manages* and *administers*. A manager gains the authority to *manage* positions by being granted it by the ultimate authority, the board. A security administrator gains the authority to *administer* a group of people by being granted it by their manager (see Figure 1).

The BOARD *grants-management* to a manager over a defined domain which he or she then *manages*:

manages (manager position) if
grants-management (BOARD manager position)

The organizational domain of the direct relation *manages* is a flat domain, consisting only of the positions defined by explicit grants of management, which are inserted into the database as facts.

Indirectly-manages is defined in terms of *manages* as follows:

position1 *indirectly-manages* position2
if
position1 *manages* position2

position1 *indirectly-manages* position1 (i.e., reflexive)

position1 *indirectly-manages* position3
if
position1 *indirectly-manages* position2
and
position2 *indirectly-manages* position3 (i.e., transitive)

There is a corresponding definition of *indirectly-contains* in terms of *contains*.

A manager *grants-admin* to a security administrator for some or all of the domain which the manager *indirectly-manages*. The use of a transitive and reflexive indirect relation permits a simple description for the organizational domain administered by a security administrator:

administers (sec-admin-name domain1)
if
grants-admin (manager-name sec-admin-position domain2)
and
occupies (sec-admin-name sec-admin-position)
and
occupies (manager-name manager-position)
and
indirectly-manages (manager-position domain2)
and
indirectly-manages (domain2 domain1)

The security administrator can then *administer* the domain which has been granted to him by the manager.

Resource authority. For the resource domain an analogous hierarchy defines the way in which a security administrator can control access rights to a resource. The main concepts are *owns* and *has-give-right* (see Figure 2).

The BOARD *grants-ownership* to a resource owner over a defined domain such as a directory, which he then *owns*. The organizational domain of the direct relation *owns* is, like *manages*, a flat domain:

owns (owner directory) if
grants-ownership (BOARD owner directory)

A manager *grants-give-right* to a security administrator for some or all of the domain which the manager *indirectly-owns*. The security administrator then *has-give-right* over the domain granted him or her by the manager:

indirectly-owns (owner directory1) if
owns (owner directory2) and
indirectly-contains (directory2 directory1)

has-give-right (sec-admin-name directory1 right)
if
grants-give-right (owner-name sec-admin-position directory2 right)
and
occupies (sec-admin-name sec-admin-position)
and
occupies (owner owner-position) and
indirectly-owns (owner-position directory2)
and
indirectly-contains (directory2 directory1)

Access rights. Access rights are granted to a position, for a particular operation on a specified directory, by a security administrator:

position-has-right (position directory right)
if
grants-right (sec-admin-name position directory right)
and
administers (sec-admin-name position)
and
has-give-right (sec-admin-name directory give-right)
and
gives (give-right right)

A person has the right to perform an operation on a data file if he or she occupies a position which has that right on a containing directory:

has-right (person data-file right) if
position-has-right (position directory right)
and
occupies (person position) and
indirectly-contains (directory data-file)

Specific policies in the model. There is no hard-and-fast distinction between the specific policies in this section and the rules in the next section. We can expect the specific policies here to remain unchanged while the organization has its present structure, but the security administrator will alter the rules in accordance with day-to-day needs.

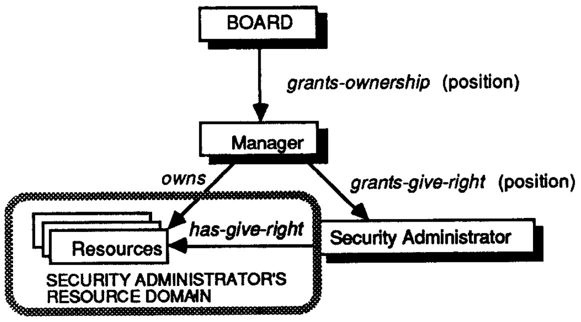
BOARD
grants-ownership (position)
Manager
owns
grants-give-right (position)
has-give-right
Resources
Security Administrator
SECURITY ADMINISTRATOR'S RESOURCE DOMAIN

Figure 2. Resource authority.

About the organization and data. The following lists facts about the organization and data. See also Figure 3 and Figure 4.

grants-management (BOARD MARKETING-DIRECTOR SALES-MANAGER)
grants-management (BOARD MARKETING-DIRECTOR DESPATCH-MANAGER)
grants-management (BOARD DESPATCH-MANAGER ORDER-SUPERVISOR)
grants-management (BOARD DESPATCH-MANAGER DESPATCH-SUPERVISOR)
grants-management (BOARD DESPATCH-SUPERVISOR DESPATCH-CLERK)
contains (COMPANY-DIRECTORY MARKETING-DIRECTORY)
contains (MARKETING-DIRECTORY SALES-DIRECTORY)
contains (MARKETING-DIRECTORY DESPATCH-DIRECTORY)
contains (DESPATCH-DIRECTORY ORDER-FILE)
contains (DESPATCH-DIRECTORY DELIVERY-FILE)

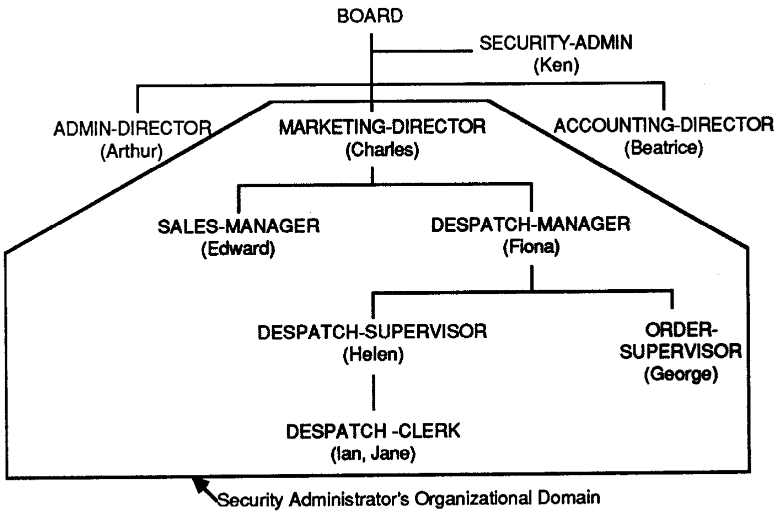
BOARD
SECURITY-ADMIN (Ken)
ADMIN-DIRECTOR (Arthur)
MARKETING-DIRECTOR (Charles)
ACCOUNTING-DIRECTOR (Beatrice)
SALES-MANAGER (Edward)
DESPATCH-MANAGER (Fiona)
DESPATCH-SUPERVISOR (Helen)
ORDER-SUPERVISOR (George)
DESPATCH -CLERK (Ian, Jane)
Security Administrator's Organizational Domain

Figure 3. Organizational structure in the model.

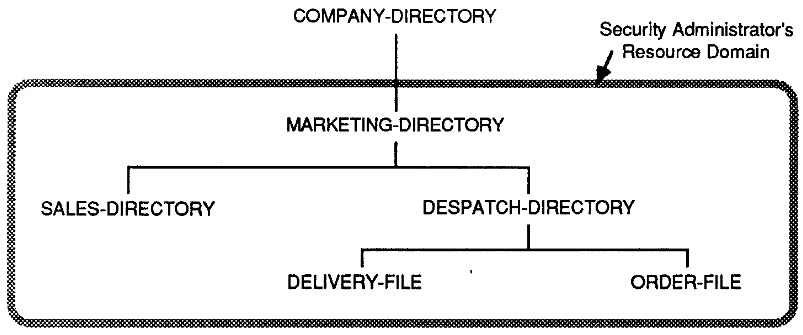

Figure 4. Structure of data used in the model.

Delegating authority.

grants-ownership (BOARD MARKETING-DIRECTOR MARKETING-DIRECTORY)

grants-admin (CHARLES SECURITY-ADMIN MARKETING-DIRECTOR)

The above rather inelegant statement expresses the marketing director (Charles) giving the security administrator administration rights for all positions in his department, including himself.

grants-give-right (CHARLES SECURITY-ADMIN MARKETING-DIRECTORY GIVE-R)

grants-give-right (CHARLES SECURITY-ADMIN MARKETING-DIRECTORY GIVE-W)

grants-give-right (CHARLES SECURITY-ADMIN MARKETING-DIRECTORY GIVE-C)

grants-give-right (CHARLES SECURITY-ADMIN MARKETING-DIRECTORY GIVE-D)

The above grants succeed because Charles, the marketing director, *owns* the marketing data.

grants-give-right (KEN ACCOUNTING-DIRECTOR MARKETING-DIRECTORY GIVE-R)

This last statement will have no effect because Ken, the security administrator, does not *own* the marketing data. He can give access rights, but not give rights. A different policy could, of course, be implemented.

Rules and facts in the model. This set of sample rules and facts can be used to run the model. Both are quite volatile. The rules are access rules made by the security administrator. The facts are descriptions of the occupants of specific positions.

Occupants of positions. The following lists the occupants of positions, as shown in Figure 3:

occupies (ARTHUR ADMIN-DIRECTOR)

occupies (BEATRICE ACCOUNTING-DIRECTOR)

occupies (CHARLES MARKETING-DIRECTOR)

occupies (EDWARD SALES-MANAGER)

occupies (FIONA DESPATCH-MANAGER)

occupies (GEORGE ORDER-SUPERVISOR)

occupies (HELEN DESPATCH-SUPERVISOR)

occupies (IAN DESPATCH-CLERK)

occupies (JANE DESPATCH-CLERK)

occupies (KEN SECURITY-ADMIN)

Granting access rights. The following lists a series of access rights granted by Ken, the security administrator:

grants-right (KEN DESPATCH-CLERK DESPATCH-DIRECTORY W)

grants-right (KEN DESPATCH-CLERK DESPATCH-DIRECTORY R)

grants-right (KEN ORDER-SUPERVISOR MARKETING-DIRECTORY R)

The above grants succeed, because the positions are within Ken's organizational domain and the data is within his resource domain. Alternatively, the attempted grant

grants-right (KEN ADMIN-DIRECTOR MARKETING-DIRECTORY R)

has no effect, because the administration director is not in Ken's organizational domain.

Sample give-right queries.

has-give-right (KEN MARKETING-DIRECTORY W) ? Yes

has-give-right (BEATRICE MARKETING-DIRECTORY R) ? No

Sample access-right queries.

has-right (IAN DESPATCH-DIRECTORY R) ? Yes

has-right (JANE ORDER-FILE W) ? Yes

has-right (GEORGE
DELIVERY-FILE R) ? Yes
has-right (ARTHUR
MARKETING-DIRECTORY R) ? No

We have discussed the need for protection in commercial organizations, and the way control principles have met this need despite having evolved before computer systems came into use. The basis for these principles is the concept of authority and how it is delegated within an organization. One of the most important of the control principles, segregation of duties, can be applied directly to the administration of access control. It implies a distinction between having access rights and being allowed to pass them on (give-rights).

The components of commercial policy are:

• *Users of the system.* Although it is people who use the system, policy decisions are normally taken with respect to organizational positions. We therefore make a distinction between people and the positions they occupy.

• *Resources*, which are the programs, services, or data accessed by users.

• *Operations* which can be performed on a resource. Access control must be applied to individual operations rather than the resource as a whole.

• *Domains*, which are the boundaries within which the policy applies. Two types of domain have been used: the organizational domain for determining the people who a security administrator has authority to administer, and the resource domain for determining the resources to which he or she can allow access. The ability to specify and delegate authority over subdomains helps in specifying policy.

• *Authority*, which is the legitimate power to implement policy. Authority is delegated from the top management of an organization and is bounded by the domains defined when it was delegated. Authority over people is bounded by organizational domains and authority over resources, by resource domains.

It is feasible to express commercial security policy embodying all these concepts within a Prolog simulation model, which specifies:

- *General policies*—the general relationship between managers, resource owners, security administrators, and the organizational and resource domains.
- *Specific policies*—the way in which a particular imaginary organization has decided to structure its domains in accordance with general policies.
- *Access rules*—the day-to-day decisions of a security administrator implementing the organization's policies.

The advantages of the Prolog approach are that it gives an executable model which can be queried to discover the consequences of the policies modeled. The backtracking used by Prolog means that we can analyze the reasons for particular results. Its main disadvantage is that writing in Prolog directly uses a very low level of specification. It is possible to validate specific cases but not prove the security of general policies. To do this, we need a higher-level formal model. The use of the formal description language Z[14] is being explored.

The model has dealt so far with a single system, a single physical domain, and a single source of authority—a comparatively simple configuration which could in practice be dealt with manually. The real incentive for executable models is for distributed and networked systems, with multiple sources of authority spread across numerous locations. The complexity of the interactions between authority domains makes it impossible to manually keep track of the distribution of access rights. Our initial work on the use of executable models has proved very promising for the validation of access control in such systems. □

References

1. R. Needham and M. Schroeder, "Authentication Revisited," *ACM Operating System Review*, Jan. 1987, p. 7.
2. K.K. Wong and W.F.N. Farquhar, *Computer Related Fraud Casebook*, BIS Applied Systems Ltd., London, 1986.
3. R.S. Waldron, *Practical Auditing*, HFL Publishers Ltd., London, 1978.
4. C.E. Landwehr, "Formal Models of Computer Security," *ACM Computer Surveys SIGOPS*, Sept. 1981, pp. 247-278.
5. B.W. Lampson, "Protection," *ACM Operating System Review*, Jan. 1974, pp. 18-24.
6. L. Snyder, "Formal Models of Capability-based Protection Systems," *IEEE Trans. Computers*, March 1981, pp. 172-181.
7. *US Dept. of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.78-STD, revised Dec. 1985.
8. *Open Systems Interconnection: Security Architecture*, ISO 7498/PDAD2, 1986.
9. D.C. Clark and D.R. Wilson, "A Comparison of Commercial and Military Security Policies," *IEEE Security and Privacy Symp.*, 1987.
10. J.H. Saltzer, "The Protection and Control of Information Sharing in Multics," *Comm. ACM*, July 1974, pp. 308-402.
11. S. Stepney and S.P. Lord, "A Formal Model of Access Control," *Software Practice & Experience*.
12. M. Sloman, "Distributed Systems Management," *Proc. IFIP TC6.4 Workshop*, North Holland Publishers, Berlin, July 1987.
13. *Micro-Prolog 3.1 Programmer's Reference Manual*, Logic Programming Associates, London, 1984.
14. I. Hayes, ed., *Specification Case Studies*, Prentice-Hall, Englewood Cliffs, N.J., 1987.

Jonathan D. Moffett is currently doing research on security in distributed systems at Imperial College, London. He is sponsored by Esso UK, having acted as the computer security adviser at Esso UK and Esso Europe Inc. for eight years. His previous experience has been as a systems consultant on large commercial systems. He has been a consultant on computer security to various banks and financial organizations.

Moffett received his bachelor's degree in mathematics in 1961 at Trinity College, Cambridge. He is a member of the British Computer Society and the Assoc. of Certified Accountants.

Morris S. Sloman has been a lecturer in the Department of Computing at Imperial College since 1976. His research interests include architecture and languages for distributed systems, distributed systems management, heterogeneous systems, and real-time systems. He has been one of the principal investigators of the various research projects funded by British Coal and SERC, which led to the development of the Conic Toolkit for Building Distributed Systems.

Sloman received his PhD in computer science from the University of Essex in 1974 and worked for GEC Computers before joining Imperial College. He is a member of the British Computer Society and the BSI Committee on Open Distributed Processing.

Readers may write to the authors at the University of London, Dept. of Computing, Imperial College of Science and Technology, 180 Queens Gate, London SW7 2BZ, UK.