Computer Communications, vol 13 no 9, pp 571-580 (November 1990)

SPECIFYING DISCRETIONARY ACCESS CONTROL POLICY FOR DISTRIBUTED SYSTEMS

Domino B1/IC/3.1
18 September 1990

Jonathan Moffett, Morris Sloman and Kevin Twidle
Department of Computing
Imperial College,
180 Queen's Gate,
London SW7 2BZ.
Email jdm@uk.ac.ic.doc, mss@uk.ac.ic.doc, or kpt@uk.ac.ic.doc

ABSTRACT

This paper discusses a proposed framework for specifying access control policy for very large distributed processing systems. These typically consist of multiple interconnected networks and span the computer systems belonging to different organisations. This implies the need for cooperation between independent managers to specify access control policy. The policy specification should permit interaction between organisations while limiting the scope of what objects can be accessed and what operations can be performed on them.

The large numbers of objects in such systems make it impractical to specify access control policy in terms of individual objects. The paper explains how *domains* can be used to group objects and structure the management of access control policy. *Access Rules* are introduced as a means of specifying the access rights between a domain of user objects and a domain of target object in terms of the permitted operations as well as constraints such as user location and time of day. The use of domains for specifying the *scope* for which authority can be delegated to managers or security administrators is explained and the issues related to implementing access rules using capabilities or access control lists are discussed.

Keywords

Access control policy, security management, authority, domains, access control lists, distributed systems.

1. INTRODUCTION

Access control is concerned with ensuring that users and processes in a computer system access computer based resources in a controlled and authorised manner. Resources must be protected from unauthorised access and access permission must be assigned only by users or managers with authority to do so. We consider only *logical* access control and not issues relating to *physical* access control such as locks, secure rooms etc. The DoD trusted computer evaluation criteria [DoD 1985] define two kinds of logical access control. *Discretionary* access control permits managers or users to specify and control the sharing of resources with other users. *Mandatory* Access Control enforces polices which are built into the design of the system and cannot be altered except by installing a new version of the system. For example in 'multi-layer' security systems, data cannot be read by a user with a lower security classification than has been assigned to the data. Mandatory policy is particularly applicable to military environments, whereas discretionary policy typically applies to commercial, industrial and educational environments. This paper is concerned with the latter.

We are concerned with very large distributed processing systems which typically consist of multiple interconnected networks and span the computer systems belonging to a number of different organisations. Authority cannot be delegated or imposed from one central point, but has to be negotiated between independent managers who wish to cooperate but who may have a very limited trust in each other. They may wish to give each other access to their computer systems which is closely controlled both in the scope of the objects which can be accessed and the operations which may be performed on them. (We use the term *object* in this paper to denote a computer resource such as a file or process as well as a user or manager).

The numbers of objects in a very large distributed system can total hundreds of thousands. It is impractical to specify access control policy in terms of individual objects and so there is a need for grouping objects and applying a policy to a group of objects. For example the same policy may apply to all people in a department or to the set of files pertaining to an application. We introduce *domains* for grouping objects in section 2 and explain their use of as a means of structuring and managing the specification of access control policy, by means of *Access Rules*, in section 3.

Access control policy relates to *authority* i.e. power which has been legitimately obtained and to how authority is delegated. We have identified distinct management roles which apply to commercial organisations, each with different authority, and in section 4 introduce the concept of *delegation of authority*, as a means of the controlled creation of Access Rules.

A number of implementation issues arise from the concept of Access Rules. They are discussed in section 5.

2. DOMAINS

Domains provide a flexible and pragmatic means of specifying boundaries of management responsibility and authority. They are described in detail in [Sloman 1989], but are summarised here. A *domain* is an object which represents a collection of objects which have been explicitly grouped together for a purpose i.e.to which a common management policy applies. The objects may be resources, workstations, modems, processes, etc, depending on the purpose for which a particular domain is defined, and each object has a globally unique identifier. A domain has an attribute called an *Object Set*, which defines the set of member objects. The minimum representation of a member object is a its unique identifier which can be used to locate the object by means of a location service. The domain may also cache object addresses and may optionally hold a local name for use by human users to refer to an object.

An object is referred to as a *member* of a domain if its identity is a member of the domain's Object Set. Objects may be a member of more than one domain at a time. Domains are persistent even if they do not contain any objects - it must be possible to create an empty domain and later include objects in it.

Domains do not encapsulate the objects themselves - managers or external objects may interact directly with an object in a domain.

Since domains are themselves objects, they may be members of other domains. An object is an *indirect member* of a domain Dx if it is a member of a domain Dy which is a member or indirect member of Dx.

In addition to the Object Set attribute, domains may have other attributes. We envisage attributes which express possible domain policies, such as constraints on domain membership. Also, some implementations may require a domain-related attribute for all objects, a Domain Set attribute recording the identities of all domains which contain the object.

2.1 Domain Relationships

The possible relationships between domains can be defined in terms of the relationships between their members as shown in Fig. 2.1. The notation is that round boxes are domains and squares or triangles are objects.

Disjoint

Two domains are defined to be *disjoint* if they have no members in common.

Overlap & Subset

Two domains *overlap* if there are objects which are members of both domains, as shown in Fig. 2.1a. A special case of overlapping occurs when the object identities in one domain's Object Set are a *subset* of those in another (Fig. 2.1b).

Subdomain

If a domain object is a member of another domain, it is referred to as a *subdomain* of that domain as shown in Fig. 2.1c. The representation of subdomains in Fig. 2.1c can lead to ambiguity between subsets and subdomains and an alternative representation is shown in Fig. 2.1d. In this paper we will be only discussing subdomains so will use the representation of Fig. 2.1c.

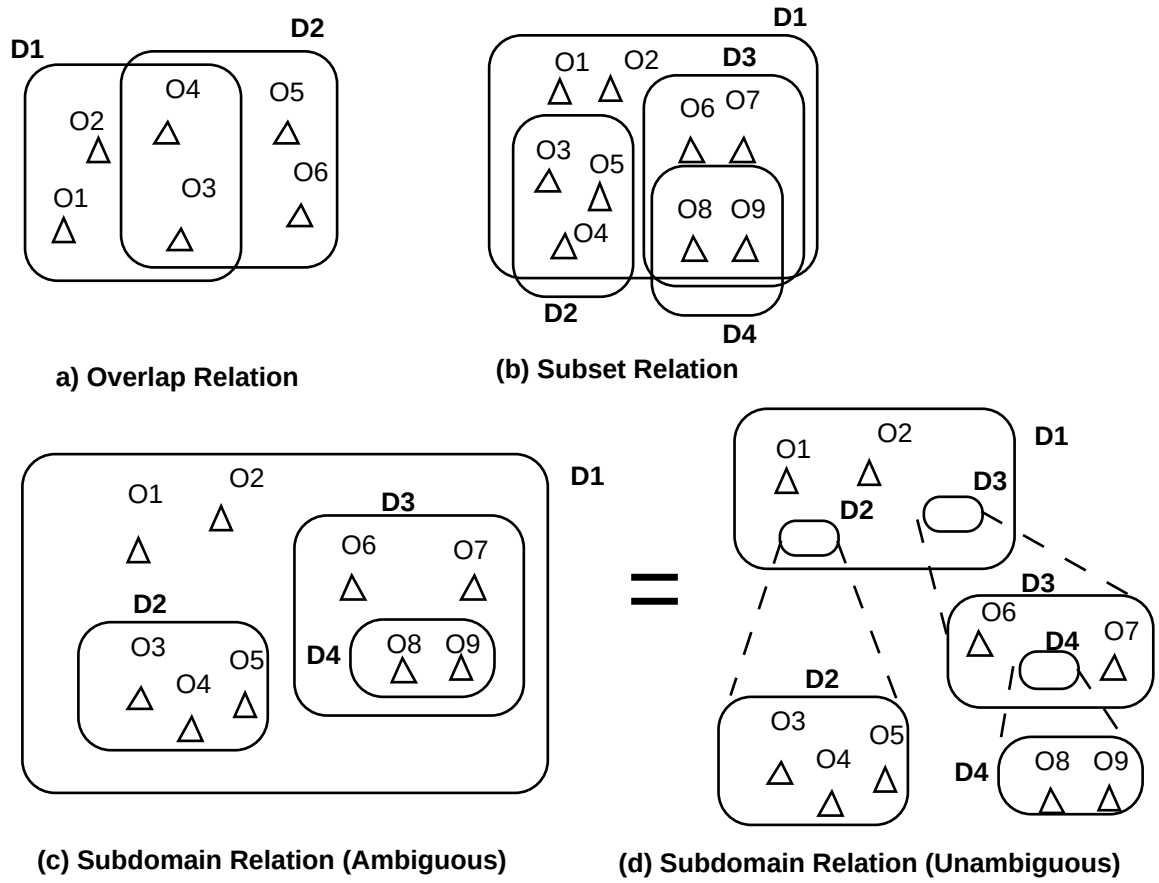


Figure 2.1 Domain Relationships

2.2 Operations on a Domain

The main operations which can be carried out on a domain are:

Creating and Destroying Objects

Create an Object in a Domain - This creates an instance of an object and adds its identity to the Object Set of a domain and is the way an object is made known to the system. Management of objects in domains requires that all objects, except for one or more root domains, should always exist in domains. Note that the Create operation may involve loading code onto a computer and creating a process as an object. In general, manufacture of an object and installing or connecting it to the distributed system will require some services outside the domain framework.

Destroy an Object in a Domain - This removes the object's identity from the domain's Object Set and destroys the object instance.

It is an application decision whether a separate Destroy operation is required, or whether it is to be achieved by Removing the object from all domains; also, whether Destroy should be permitted on an object which is a member of more than one domain. These policy decisions can be represented as Domain constraints.

Operations Relating to Domain Membership

Include Object in a Domain - includes an object into a domain by adding its identity to the Object Set of the domain. It does not affect the state of the object or its membership of other domains, except that the object updates its Domain Set. A domain object can be included into a domain to form a sub-domain.

Remove Object from a Domain - removes an object from a domain by removing its identity from the Object Set. There is no effect on the state of the object except that if it maintains a Domain Set, this must be updated.

It is an application decision whether Remove is permitted if it is not a member of any other domain's Object Set; the effect is then to destroy it.

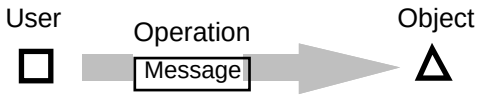
List Objects in a Domain - returns a list of the members of the Object Set of a domain.

3 ACCESS RULES

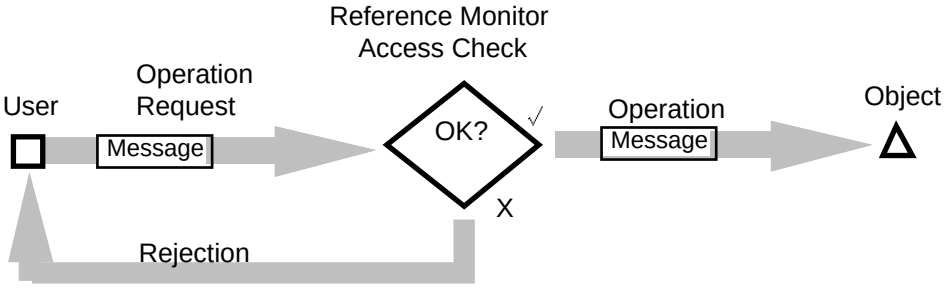
3.1 Access Control Enforcement

We use the *Reference Monitor* concept, as described in [Anderson 1972, DoD 1985], to model an access control system. The function of a reference monitor is to enforce the authorised access relationships between subjects (users) and objects of a system. It is a trusted component of the system. All operations requested to be carried out on an object are intercepted by the reference monitor, which only invokes the operation if the access is to be allowed. See Figure 3.1.

The figure illustrates that, in general, access control is carried out by the system, and not by the target object itself, although the reference monitor could be implemented by the object. The operation of the Reference Monitor is transparent to the user provided the access is authorised, but if the access is not authorised the operation is not invoked and the user is informed by a failure message.



a) User View of an Operation



b) Access Checking by a Reference Monitor

Figure 3.1 Reference Monitor Access Check

3.2 Motivation for Access Rules

Access Rules are the means for specifying access control policy and are an adaptation of Lampson's Access Matrix [Lampson 1974] for large distributed systems. Figure 3.2 shows a portion of an Access Matrix which indicates, for each user-target pair in the system the operations which the user may perform upon the target object. The default access is zero if none is shown.

Target Objects / Users	**Obj1**	**Obj2**	**.....**	**ObjN**
User1	Read	Read, Write	**.....**	Read
User2		Write	**.....**	
.....				
UserN	Read		**.....**	Read, Write

Figure 3.2 Portion of an Access Matrix

The Access Matrix is unsuitable for large distributed systems for several reasons:

i) It becomes too large when the number of objects can be in the hundreds of thousands. Sparse matrix storage mechanisms can alleviate the problem to some extent, but it is necessary to partition the matrix. Domains provide a possible method of partitioning.

ii) The matrix, as a centralised data structure, assumes global knowledge of the user and target objects which are in the system, but it is impractical to maintain this global information in a distributed system spanning multiple independent organisations, so the full matrix can never be constructed.

iii) The access matrix is based on the assumption that access rights are specified between individual users and target objects. It does not permit policy to be specified with respect to groups i.e. to domains. The manager may not even know the members of the domain at the time the policy is formulated, and the policy specification should not require changes when the individual user and target objects in the system change.

iv) We require a means of specifying policy in terms of roles performed by users in an organisation rather than the users themselves. For example the access rights for a departmental manager should not relate to the person holding that position, but rather to the position or role itself. This means the rights do not have to be changed if the person is moved to another role.

3.3 Definition of Access Rules

A simple Access Rule is shown in Figure 3.3. It specifies a *User Domain,* which identifies the set of possible users who can perform operations; the *Target Domain*, which identifies the set of possible target objects on which operations can be performed, and the *Operation Set* , which are the authorised operations a user can perform on a target. This may be a subset of the operations which the target objects support. Extensions to the simple access rule are discussed in section 3.6.

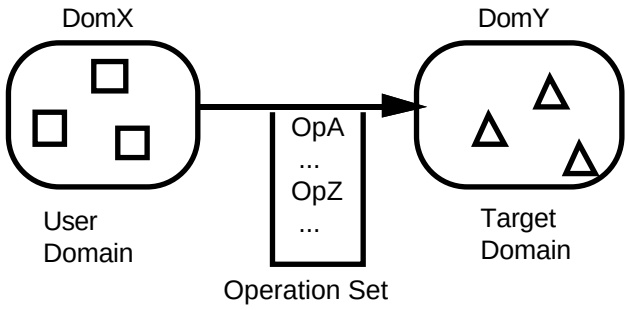


Figure 3.3 An Access Rule

In our model, access rules are the means by which the Reference Monitor determines whether to authorise an *operation request*, which is a triple consisting of (user, target, operation name). We assume that an authenticated user identity is available to the monitor as part of the message which requests the operation. The monitor authorises a request if an access rule exists which applies to the operation request i.e. if the user object is in the set defined by the user domain of the rule, the target object is in the set defined by the target domain of the rule and the operation is in the operation set of the rule, as indicated in Figure 3.4. There may also be extensions to access rules to allow for decisions based on time of request, user location and operation parameter values.

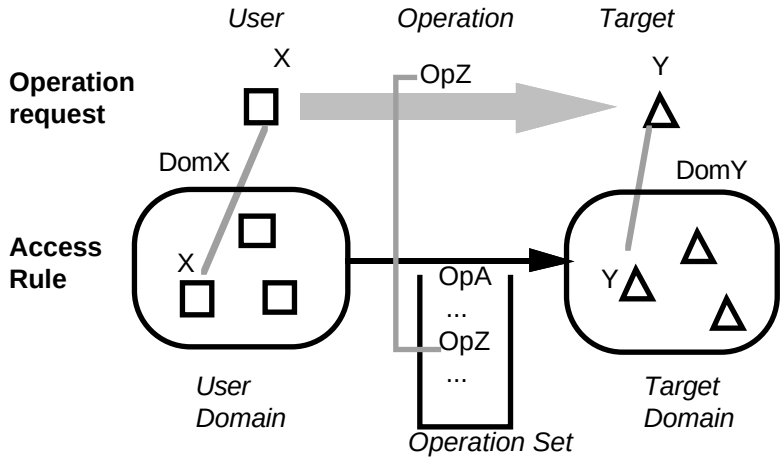


Figure 3.4 An Access Rule which Applies to an Operation Request

Objects in a subdomain, by default, “inherit” the access rules applying to direct and indirect parents. In Fig. 3.5, objects in domain D2 inherit the access rule applying to the parent domain D1 so can perform operations (OpA & OpB) on objects in domains D3 and D4 as the latter is a subdomain of D3. There may be more than one access rule which satisfies a particular request, e.g. as a result of an object's membership of overlapping domains. This indicates that authority has been granted via more than one route.

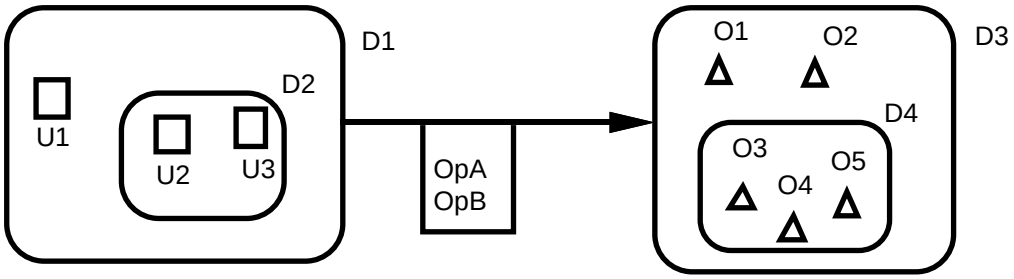


Figure 3.5 Inheritance of Access Rules

Some operations affect more than one object which implies the need for multiple access rules to authorise the operations. For example the operation to include an object in a domain requires an access rule for the object to be included as well as one for the domain into which it is to be included.

3.4 Granularity of Access Rules

The User Domains in access rules are role domains which have individual users as members. This ensures that the rules remain valid even if users change their roles in an organisation. However, there will be a requirement to set up access rules for an individual user, e.g. to allow the user to access his personal files, electronic mail, etc. Defining an access rule for an individual target object can be achieved by creating a domain and including the target object in that domain, but this would be laborious if different access rules are required for many individual objects. Our general model of domain expressions therefore allows for the specification of individual users and target objects. There may be some implementation costs associated with this, and other projects place restrictions on the facility. The Andrew project [Satyanarayanan 1989] allows individual users, as well as domains, to be specified in their Access Control Lists (ACLs), but the finest granularity for the ACLs of file objects is the Directory.

3.5 Example of Access Rules

We use as an example the Payroll Department of an organisation. It consists of roles for people, and a directory of payroll files, see Figure 3.6. We assume that the Payroll Manager (viewed as being outside the department he runs and not shown in Fig. 3.6) can create whatever Access Rules he wishes.

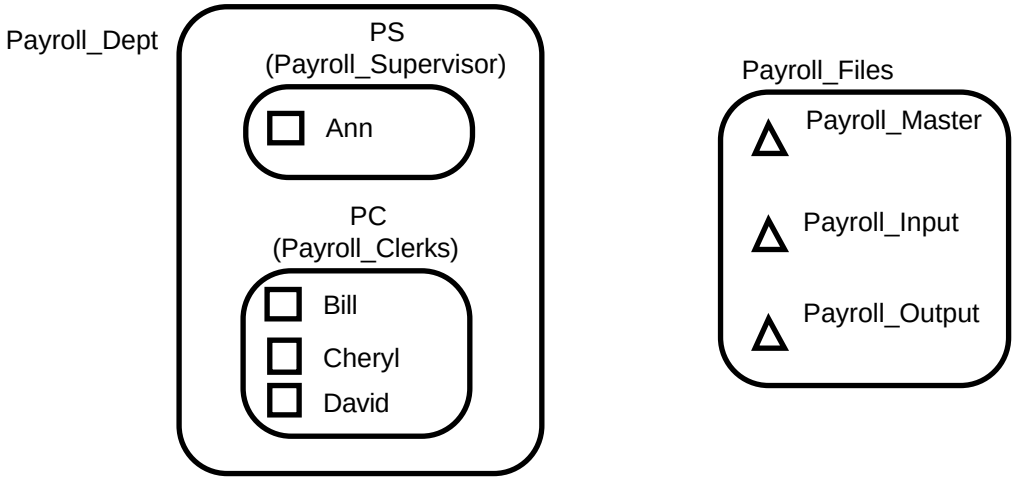


Figure 3.6 Domain Structure of the Payroll Department

The Payroll Manager may not know or understand the names of the files in the Payroll_Files Domain or what are the names of the files which will be added to it in future. However, he has delegated to the Payroll Supervisor the task of maintaining the files, so an expression of this delegation policy is to give the Payroll Supervisor the ability to Create, Read and Write files in

that Domain. His policy is also to allow the whole Payroll Department to Read the Payroll_Files. These policies are expressed by the two Access Rules illustrated in Figure 3.7.

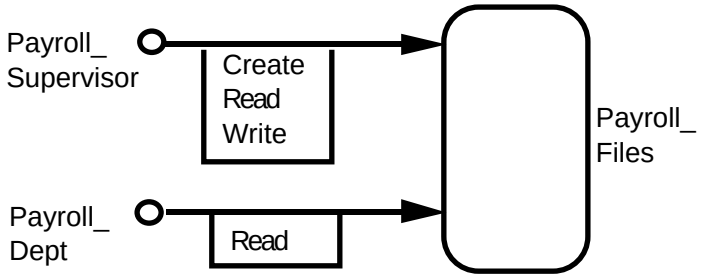


Figure 3.7 Example Access Rules

The access rules do not name the user or target objects, and so we cannot tell directly from them whether the access by a particular user to a particular target will be allowed. To interpret the access rule we have to know the domain structure, as shown in Figure 3.6, which can be used to derive a partial access matrix as shown in Figure 3.8.

Target Objects / Users	**Payroll_ Master**	**Payroll_ Input**	**Payroll_ Output**
Ann	Create, Read, Write	Create, Read, Write	Create, Read, Write
Bill	Read	Read	Read
Cheryl	Read	Read	Read
David	Read	Read	Read

Figure 3.8 Partial Access Matrix Derived from Access Rules and Domain Structure

Even if the Access Rules do not change, the matrix will cease to be a valid and complete description of the management access policy as soon as the membership of any of the Domains changes. For example, if Cheryl is replaced by Charles as a member of the Payroll_Clerks Domain or Payroll_Print is introduced into the Payroll_Files Domain, a new matrix would have to be to be constructed, but the access rules shown in Figure 3.7 would still be valid.

3.6 Access Rule Extensions

The simple access described so far specifies users or targets in terms of a domain name. However there is a need to specify more complex sets of objects in terms of a general domain expression which allows for the specification of individual users and target objects, set operations, restrictions on the interpretation of indirect domain membership and constraints on the applicability of access rules.

a) Set expressions

Domains and subdomains are a powerful means of expressing hierarchical membership and set union. If domain DomA contains DomB and DomC, then an Access Rule applying to DomA applies to the union of the Object Sets in DomB and DomC. However, it provides no means of expressing other basic set operations such as Set Difference and Set Intersection. Set Difference can express groups of objects such as 'all users in Payroll Dept except the Payroll Clerks', and 'all files in Payroll_Files except Payroll_Master'. Set Intersection can express groups such as 'all files which are in Payroll_Files and also in Personal_Data' (where we assume that Personal_Data is a domain of files subject to the Data Protection Act).

It would be possible to use domain manipulation operations to create a new domain with the required membership e.g. DomA = (DomB ∩ DomC), and refer to DomA in the access rule. However, the access rule then applies to membership of DomB and DomC at the time DomA is created and not at the time the access rule is checked. Static enumeration of objects at some point in the past may not be what is required. Set operations in access rules would be evaluated at the time the rule is checked.

b) Limiting Indirect Domain Membership

In the normal case an object is interpreted as being a 'member' of a domain if it is either a direct or an indirect member of it . This uses the full power of subdomains and is likely to be used in most cases. However, there may be situations in which we wish to prevent an Access Rule applying to indirect members of a domain. Therefore we need a notation which allows us to specify in an Access Rule that domain 'member' should be interpreted only as a direct member e.g. by suffixing the name with an exclamation mark.

c) Constraints in Access Rules

Another extension needed to the simple access rule is to permit the expression of a constraint on its applicability in terms of time of day, date, location of user etc. For example, it is quite common in many organisations to permit access to various objects only during normal office hours or to set up an access rule which expires after some time or only becomes valid in the future. Access to some resources may only be permitted to a particular user such as a salesman who is using a local terminal but when he is using a terminal at a customer's site he will have access to only a restricted set of resources or services.

We assume that the Reference Monitor has available information about the User's environment (date and time of request, source of origin, etc), and the parameters of the operation.

The possible general constraints in Access Rules, their method of representation and their implementation are all subjects for further study.

d) Logging Switch in Access Rules

A Security Audit Facility requires logging of both authorised and unauthorised operations. It is assumed that the Reference Monitor will provide it on all unauthorised operations, but there may also be a selective need for information on authorised operations also. For example all changes to access rules performed by a security administrator should be logged. One means of controlling this would be by a logging switch in Access Rules, which could be turned On in order to trigger the Reference Monitor to provide the Security Audit Facility with input. Adequate controls are required to prevent unauthorised switching Off.

3.7 Removal of Access

The prompt removal of access authority when necessary is an essential aspect of access control. There are a number of methods of achieving this:

a) It is simple to express the exclusion of a particular user from accessing a target domain at the time an access rule is set up by means of a domain expression such as (Users_X \ Users_Y), which excludes members of Users_Y from access which they would have been permitted as members of Users_X. Note, however, that this will not override another Access Rule which allows members of domain Users_Y to have access. Similarly, a Target domain expression may specify (Files_X \ Files_Y).

b) Negative Rights, which override any positive rights, have been used in some systems as a means of rapidly and selectively revoking access to sensitive objects [Satyanarayanan 1989]. This greatly complicates the semantics of an Access Control system and can introduce inherent contradictions [Tygar 1987]. We therefore conclude that they should not be introduced into our model and so we only permit positive authorisation.

c) Destruction or modification of an access rule will not prevent access if there is another remaining rule which permits access. Determining which access rules permit access can be complicated because typically the user or target object is a member of several domains which are subdomains of ones to which access rule refer. Tools and reporting facilities are therefore

needed to permit a security administrator to determine what target domains a user can access and what users can access a particular target domain. The use of these tools would be adequate for routine removal of access rules, but searching a large system for relevant access rules may be rather slow.

d) Denying access to an individual user in an emergency should be performed not by alteration of Access Rules but by suspension or deregistration of the user. It is instant and effective and does not add complications to the system.

There are thus a number of different strategies which can be applied depending on the circumstances.

4. DELEGATION OF AUTHORITY

4.1 Management Roles

Access control policy relates to *authority* i.e. power which has been legitimately obtained and to how authority is delegated. We have identified four typical roles, which apply to commercial organisations, each with different authority:

i) An *owner* of an object has the legal power to perform any feasible operation on it (including giving away ownership), together with the responsibility for operations which are performed on the object. There are of course limitations to this power. For example an owner of personal data may not disclose it except under the terms of the Data Protection Act, and the owner of petroleum spirit at a bulk distribution plant must ensure that an automated system dispensing it to tanker lorries does so in accordance with certain safety regulations. Restrictions of this kind have to be represented in computer systems as mandatory constraints, and are beyond the scope of this paper.

   Ownership could be shared or held by a committee or a board of directors. Note that the owner is not necessarily the user who creates an object, just as the bank clerk who creates an account for a customer is not the owner of the account.

ii) A *manager* has been delegated management authority for objects by the owner. In general managers can perform any operations, including management operations, on an object but may have some limitations on their authority. For example a manager may not be able to pass on his own management authority.

iii) In many commercial organisations a policy of separation of authority applies. The job of issuing access rights to users is delegated by an owner or manager to a *security administrator*. However the security administrator does not usually have access rights to the target objects for which he/she gives rights.

iv) Normal users are given authority to use objects but not to manage them. A user cannot normally delegate any authority which he holds.

We envisage different organisations defining other roles to suit their particular structures. The roles are specified as domains which define the authority pertaining to the users which are members of the domain.

4.2 Scope of Authority

Access permissions are given and removed by creating, destroying and modifying access rules. The operations Create, Destroy and Read are needed on access rules. A Modify operation could be defined using a compound of these.

Authority is delegated by creating access rules for manager or security administrator domains. However it is still necessary to control the flow of authority. A security administrator must be prevented from creating an access rule for users or targets over which he has no authority. Thus owners, managers, and security administrators have a *user scope* which is a domain of users to whom they can delegate authority and a *target scope* which specifies the target domains to which they are permitted to delegate authority.

For example a Security Administrator domain has two attributes, a SA_User_Scope which defines the users who are administered, and a SA_Target_Scope which defines the target objects to which he can give access. He can create access rules to allocate access rights within

these scopes. We can achieve separation of authority by ensuring a Security Administrator is not a member of the user scope over which he has authority. He cannot then give himself access rights.

Users also need to specify access rules to permit colleagues etc. to access resources which are considered personal to them, for example to share files for cooperative working. One solution would be to consider a user to have the combined role of security administrator of a personal domain - see section 5.6.

In addition to specifying Access Rules, managers and users need to be able query the system, within their scope of authority, to determine what access is permitted. Managers and security administrators need to be able to analyse the system to determine inconsistencies such as inaccessible domains. Typical reports required for managing and auditing the security policy include:

- All objects to which selected user have access
- All users who can access selected objects
- All failed access attempts
- Selected logging of authorised accesses.

5. IMPLEMENTATION ISSUES

The above section has described Access Rules as a means of specifying access control policy. This is essentially being proposed as the user view of security policy, which must then be reflected in an underlying implementation which makes use of access control mechanisms to implement the policy.

The main implementation issues, which should be hidden from users, are:

- How to store Access Rules - whether they should be associated with the user objects, target objects or independent of both.
- How to implement the reference monitor in a distributed system.
- How to translate from the user view (access rule) to the implementation mechanism being used.

5.1 Implementation Requirements

a) Validity

It is essential that the underlying implementation should make access control decisions which are consistent with the user view of domain-based Access Rules. When an Access Rule is created, modified or destroyed, the effect of the change must be reflected in access control decisions. It may be permissible in some circumstances for the effect to be delayed (e.g. until the following day, in the case of non-urgent changes), but this must be predictable. In addition when an object is included in or removed from a domain, or a new object is created, then the object's access authorisations (either as user or target) which derive from membership of the domain must change accordingly.

b) Performance

The overhead introduced for validation of authorised operation requests must be minimal if the system is not to degrade.

The overhead in updating access authorisations must be tolerable for domain membership changes. It is thought that the contents of user domains will change relatively infrequently compared to those of target domains which contain objects such as temporary files.

Changing access control policy by updating access rules, or obtaining status reports will be relatively infrequent and do not always have to take immediate effect, so higher overheads can be tolerated.

5.2 Representing Individual Users

In our proposed implementation, an individual user is represented in the system by a "login" domain with associated access rules which define the authority of that user. When the user logs into the system a user interface object is created to perform operations on her behalf. We have been using the term User to mean this login domain.

In addition the user has a personal domain (c.f. home directory) over which she has complete authority. The user can create sub domains to hold objects such as files and to which she can permit other user to have access i.e. the user is also a security administrator with unlimited SA_User_Scope and a SA_Target_Scope referring to the personal domain.

A person can take on a particular role, such as one of those discussed in section 4.1, by moving the user (domain) into the role domain and so the access rules of the role domain apply to the user.

5.3 Implementation Options

a) Reference Monitor Implementations

When considering implementation issues it is useful to refine the reference monitor shown in figure 3.1, into two components - an Access Control Decision Facility (ADF) and an Access Control Enforcement Facility (AEF) as in the OSI security framework [ISO 1989]. However the OSI security model assumes that the AEF always calls the ADF, which is not the case as shown below.

ADF Associated with Target Object

The ADF may be in the same trusted computer or system as the target object so that communication is secure. In that case the implementation may be as defined in [ISO 1989], and shown in Fig 5.1. Since communication is secure, it may be assumed that the message has not been interfered with between leaving the ADF and reaching the object. Many mainframe access control systems such as IBM RACF [IBM 1986] follow this approach.

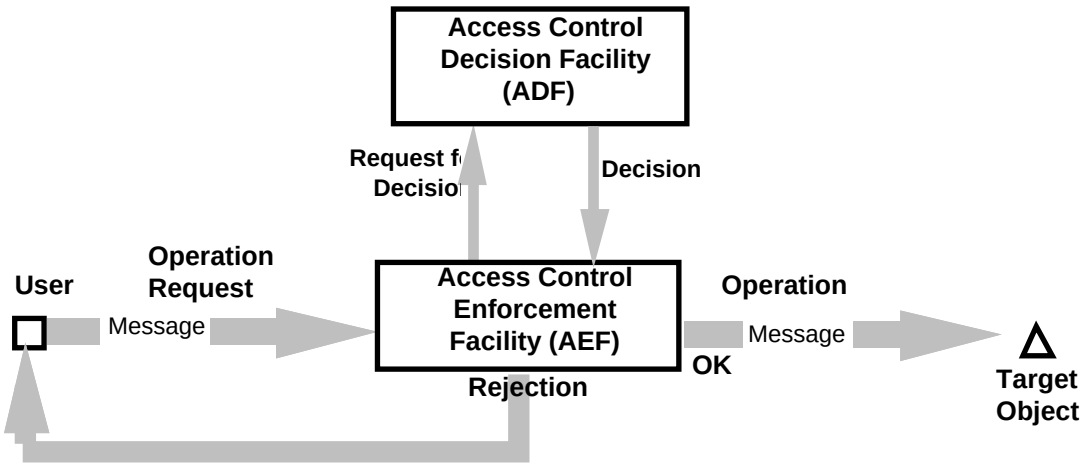


Figure 5.1 ADF Associated with Target Object

ADF Generates Capabilities

The ADF may communicate with the target object via an insecure transmission medium which implies the message may have been interfered with between leaving the ADF and reaching the target object. It is therefore necessary to generate a secure capability, as is done in Kerberos [Steiner 1988], which is checked by the AEF close to the target object (see Fig 5.2). The capability mechanism separates the ADF which generates the capability from the AEF which enforces it, and is therefore a suitable mechanism for a distributed system, in which one cannot use a single centralised reference monitor mechanism.

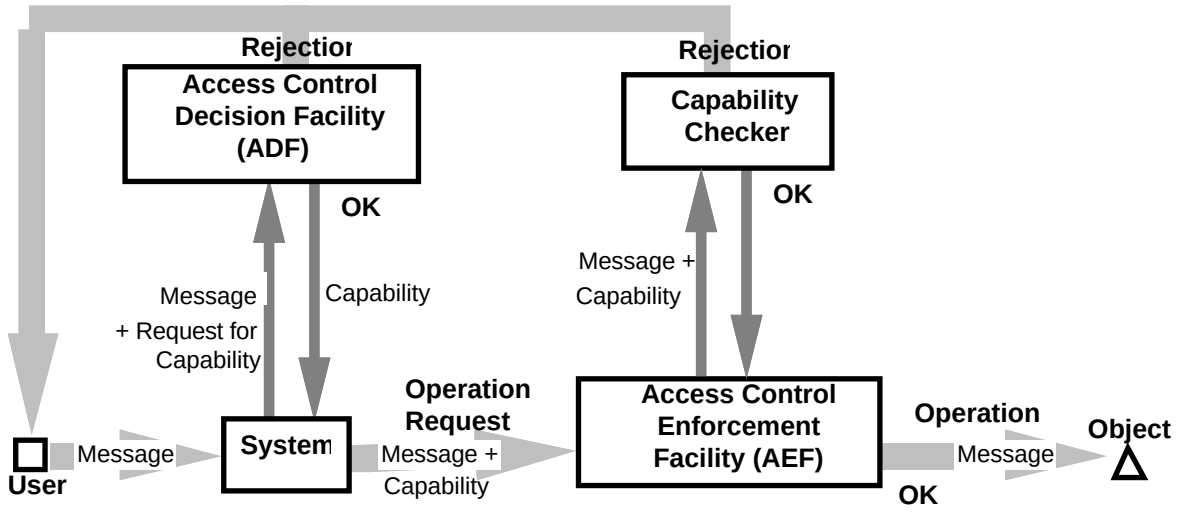


Figure 5.2 ADF Generates Capabilities

Capabilities can be generated, when a session is established between a user and a target domain and or even when the user logs into the system. They can be cached at the user and reused for multiple operations, thus reducing the overheads of operation invocation to that of checking capability validity.

b) Location of Access Rules

Access rules can be held with the users, target objects or independently in a rules database.

Access Rules Database

An access rules database can be independent of both user and target domains as part of an authorisation service. It would have to be based on a distributed database and distributed authorisation servers. It would have to hold information about domains and their relationships and so would lead to a duplication of information in the system. Therefore it makes sense to relate the access rules to the domain objects which would already be distributed.

Access Rules Held with User Domains

Access rules can be held as attributes of User domain objects. The access rules (and ADFs) would thus be distributed to reflect the distribution of the user domains and the ADF shown in fig 5.2 could be co-located with the domain, so generating a capability could be performed at the relevant domain object. The problem with holding access rules with the user domains is that they may exist in personal workstations which cannot be trusted, and their security is therefore difficult to enforce. In addition access rules are more likely to be generated by a security administrators in the organisation of a target domain rather than user domain

Access Control Lists (ACLs) with Target Domains

An alternative approach is to hold access rules as ACLs which are attributes of the target domains. The security of the access rules can then be commensurate with the security of the objects they are protecting. Target objects are more likely to be based on servers, under the control of system managers rather than individual users and their domains and access rules can be held on more secure systems.

The target object and its domain may be on different computers. If an access control decision takes place only when a session is established between a user and target object, then the overheads of a remote access by the AEF to the ADF co-located with the target's domain would be acceptable. However if access checking is required on every operation invocation then remote access to the ADF would be unacceptable and every target would have to hold its own ACL. This requires a "flattening" of ACLs whereby an ACL inherited from a parent domain is

incorporated into a nested domain and every object holds a copy of the ACL of the domains it is in.

A better solution is to combine the ACL and capability approach with a capability being generated by the ADF from an access rule held with the target domain when a user-target session is established. The capability is held by the user and can be checked by the target for each operation invocation.

The disadvantage of a pure ACL approach is that it can lead to a breach of security in systems with nested operations. If a server object has to invoke operations on other objects on behalf of a user it usually adopts the identity of the user in order to obtain user-specific authorisation for the operation. It has been pointed out by [Vinter 1988] (among others), that this breaches the 'least privilege principle' as the server can now perform all the operations authorised for the user, and not only the one which was requested. This only works when servers are trusted, which may not be the case if the server belongs to another organisation on a remote network.

5.4 Implementation Approach

We are experimenting with an implementation of access rules based on ACLs stored with domain objects [Twidle 1988]. Our testbed is the Conic toolkit for building distributed systems [Magee 1989]. The access control decision can be made when an interface of one object is bound to the interface of another which is analogous to setting up a session between them. In Conic an interface is made up of a set of ports where a port represents an operation which can be performed on the object. Binding an interface on a user object to that on a target object can also permit the target object to invoke operations (i.e. back calls). Binding can be selective for individual ports, if an access rule specifies a subset of the permitted operations. The use of ACLs does imply a directionality in the binding (from the user to the target), but not on the use of the interfaces.

Each object will hold information on the domains of which it is a direct member. When a binding takes place the list of direct parent domains of the user object is passed in the binding request sent to the domain of the target object. It searches its ACL for a matching rule. If none is found there will be a search upwards through the domain structure for an entry in a parent domain's ACL which will authorise the request. Fig. 5.3 shows an object in Domain B which is able to access an object in Domain D because Domain C's ACL has an entry for Domain A.

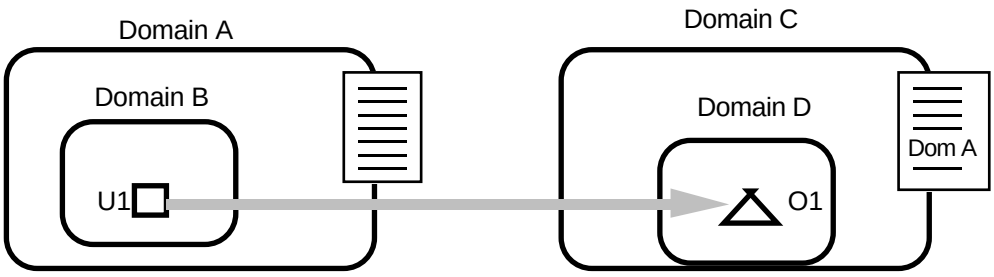


Figure 5.3 Access Control List Example

When U1 is bound to O1, the bind request contains U1's parent domain (B). Domain D's ACL is searched for an entry for B. None is found so a list of Domain B's parents is obtained and D's ACL is searched again. As this fails Domain C's ACL is searched for an entry for B then A and an entry for the latter is found so the binding succeeds.

The above search may require locating indirect parent domains of the user object which may be at a different site. For practical purposes we intend to place an arbitrary limit on the levels of nesting for "inheriting" access rules to limit the search path for a match. We currently have a distributed name server which is being extended to act as a distributed domain server and hold the access rules and domain hierarchies.

6. RELATED WORK AND CONCLUSIONS

6.1 Related Work

The use of an access matrix which incorporates domains is described in [Linden 1976]. In our terminology, these domains correspond to a set of access rules for a particular user and hence define the access rights for a single user.

In the field of access control, the greater part of the work has been concentrated on Mandatory Access Control in order to achieve a multi-level secure system as defined in [DoD 1985]. Relatively little has been done in the area of discretionary access control. [Snyder 1981, Stepney 1986] deal with access control to individual objects, but do not consider the problems of structure and scale. An exception to this is [Robinson 1988], which introduces the concept of Domains as a means of managing systems and uses access control as one of its applications. His work was relatively implementation-specific, and one of our motivations has been to generalise it for use as a management tool.

There are a number of groups in the USA working on security management who also use the concept of a domain for distributed system management. Estrin's inter-organisational networks [Estrin 1987] correspond roughly to a physical domain with a common set of access rights. [Nessett 1984] introduces the concept of separate source, registration and destination domains for authentication. This is developed by [Gomberg 1987], as a means of controlling logon across distributed systems. Kerberos [Steiner 1988] is an authentication service which has a concept of *realm*, corresponding to Nessett and Gomberg's registration domains. All of the above combine authentication with a coarse-grained authorisation service; the target objects are networks, computers or services, so the number of objects is relatively small.

Little also has been done on the way in which authority can be transmitted in large organisations. [Lampson 1974] and [Snyder 1981] make explicit assumptions that the authority to give access rights is bound in with the access rights themselves. Stepney and Lord [Stepney 1986] recognise that it is separate, but explicitly regard it as being outside the scope of their model.

6.2 Further Work

There are several areas of work which require investigation:

- Our ACL implementation of access rules is still at a very early stage and so we need practical experience to determine whether performance is adequate.
- We believe that an ACL based implementation in which checking only takes place at bind time is adequate for many commercial purposes. However we intend to investigate generating capabilities at bind time to permit checking for every operation invocation.
- Reporting and monitoring facilities are an essential part of access control for any real system, and their design requires further work.
- Our work is currently targeted at discretionary access control, but at some point it will be necessary to consider how it would integrate with mandatory access control systems.
- We are working on a framework for specifying and controlling the delegation of authority from owners, through managers, to security administrators [Moffett 1990].

6.3 Conclusions

The paper has shown the use of domains and access rules as a means of specifying discretionary access control in a way which promises to meet the needs of large scale distributed systems. The ability to group objects into a domain and specify a single access rule which applies to all the objects in the domain is one approach for catering for large scale. The concept of a subdomain inheriting an access rule of a parent permits access control policy to be specified at a very coarse grain for some purposes but a particular access rule can be set up for a small subdomain where appropriate. This gives considerable flexibility for specifying access control policy which applies to the many different sets of objects which would occur in a large distributed system.

The work described in this paper is being undertaken as part of DOMINO project on Domain Management in Open Distributed System (Project No. IED 3/1/1409). This is collaborative project funded by the U.K. Information Engineering Directorate and involves Imperial College, SEMA Group and British Petroleum. This project is concerned with managing large scale distributed systems and the use of access rules will permit the specification of management policy in terms of what management authority is delegated to a domain of managers. The concepts are also being used to specify responsibility and authority for configuration management in an environment such as a factory.

7 ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the IED and the SERC ACME directorate (Grant No. GR/E 62394). Our colleagues Jeff Kramer, Jeff Magee, Naranker Dulay & SC Cheung of Imperial College, John Haberfield & Tony Law of BP and Tony Jeffree of Sema Group have contributed to the concepts described here. Thanks also to the anonymous referees for their suggestions for improvements to the paper.

8. REFERENCES

[Anderson 1972] Anderson J.P., Computer Security Technology Planning Study, ESD-TR-73-51, vol 1 AD-758 206, ESD/AFSC Hanscom, AFB Bedford, Mass, Oct 1972.

[DoD 1985] Department of Defense (USA), Department of Defense Trusted Computer System, Evaluation Criteria, DOD 5200.78 - STD, Dec 1985.

[Estrin 1987] Estrin D., Controls for Interorganization Networks, IEEE Transactions on Software Engineering, Vol SE-13 no 2 (Feb 1987).

[Gomberg 1987] Gomberg D.A., A Model of Inter-Administration Network User, Authentication & Access Control, Mitre Corporation, Washington C3I Ops, 7525 Colshire, Drive, McLean VA 22102, MTR-87W00003, Dec 1987.

[IBM 1986] OS/VS2 MVS RACF, 5740-XXH, General Information (IBM Technical Manual GC28-0722) .

[ISO 1989] ISO, Security Framework III: Access Control Framework, ISO/IEC JTC1/SC21 N4206, Nov 1989.

[Lampson 1974] Lampson B.W., Protection, ACM Operating System Review, Vol 8 no 1, Jan 1974, pp 18-24.

[Linden 1976] Linden T., Operating System Structures to Support Security and Reliable Software, ACM Computing Surveys, Vol. 8, No. 4, Dec. 1976, pp. 409-445.

[Magee 1989] Magee J., Kramer J. & Sloman M.S., Constructing Distributed Systems in Conic, IEEE Transactions on Software Engineering, vol 15 no 6, June 1989, pp 563-575.

[Moffett 1990] Moffett J.D. & Sloman M.S., Delegation of Authority, Domino paper B1/IC/4. 19 July 1990.

[Nessett 1988] Nessett D.M., The Inter-Authentication-Domain (IAD) Logon Protocol, (Preliminary Specification and Implementation Guide), Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550, 6 May 1988.

[Robinson 1988] Robinson D.C. & Sloman M.S., Domain Based Access Control for Distributed Computing Systems, IEE Software Engineering Journal, Sept 1988, pp. 161-170.

[Satyanarayanan 1989] Satyanarayanan M., Integrating Security in a Large Distributed System, ACM Transactions on Computer Systems, Vol 7, no 3, Aug 1989, pp 247-280.

[Sloman 1989] Sloman M.S. & Moffett J.D., Managing Distributed Systems, Sept. 1989, Submitted for Publication.

[Snyder 1981] Snyder L., Formal Models of Capability-Based Protection Systems, IEEE Trans on Computers, Vol 30 no 3, March 1981, pp 172-181.

[Steiner 1988] Steiner J.G., Neuman B.C. & Schiller J.I., Kerberos: An Authentication Service for Open Network Systems, Winter Usenix 1988, Dallas, TX.

[Stepney 1987] Stepney S. & Lord S.P., A Formal Model of Access Control, Software - Practice & Experience, vol 17, no 9, Sept 1987, pp 575-593.

[Twidle 1988] Twidle K. & Sloman M.S., Domain Based Configuration and Name Management for Distributed Systems, IEEE Workshop on Distributed Computer Systems in the 1990s, Hong Kong Sept 1988.

[Tygar 1987] Tygar J.D & Wing J.M., Visual Specification of Security Constraints, IEEE Workshop on Visual Languages, Linkoping, Sweden, Aug 1987, pp 288-301.

[Vinter 1988] Vinter S.T., Extended Discretionary Access Controls, IEEE Symposium on Security and Privacy , April 1988, Oakland, CA, IEEE Computer Society Press, pp 39-49.