9

Modeling Mandatory Access Control in Role-Based Security Systems

Matunda Nyanchama
#23, 463 Platts Lane, London Ontario, N6G 3H2, Canada. email:
matunda@csd.uwo.ca
Sylvia Osborn
Department of Computer Science, University of Western Ontario, London, Ontario N6A 5B7, Canada. Fax: (519) 661-3515. email:
sylvia@csd.uwo.ca

Abstract

This paper discusses the realization of mandatory access control in role-based protection systems. Starting from the basic definitions of roles, their application in security and the basics of the concept of mandatory access control, we develop a scheme of role-based protection that realizes mandatory access control. The basis of this formulation develops from the recognition that roles can be seen as facilitating access to some given information context. By handling each of the role contexts as independent security levels of information, we simulate mandatory access by imposing the requirements of mandatory access control. Among the key considerations, we propose a means of taming *Trojan horses* by imposing acyclic information flow among contexts in role-based protection systems. The acyclic information flows and suitable access rules incorporate secrecy which is an essential component of mandatory access control.

Keywords

Security level, information flow, mandatory access control, role, role-based protection.

1 INTRODUCTION

Role-based security provides a flexible means of managing large numbers of access rights, especially for large database systems. A role is defined in terms of *privileges* where a privilege is a *unit* of access to system information. A role is a named collection of such privileges (Baldwin (1990), Krishnamurthy & McGuffin (1992), Nyanchama & Osborn (1994a)). User-role authorization grants the user access to the role's privileges.

Role-based protection eases the administration of privileges due to the flexibility with which roles can be configured and reconfigured (Ting, Dermurjian & Hu (1992), Nyan-

D. L. Spooner et al. (eds.), *Database Security IX*

chama & Osborn (1994a)). Privileges can be assigned or revoked from a given role thus providing for an incremental manner of administering access rights. With roles, we can enforce the principle of *least privilege* where a role is assigned only sufficient functionality to realize the intended duty requirements (Thomsen (1991)).

From an authorization point of view, role-based security designates what operations can be invoked via a given role and hence what information can be accessed via the role. A role specifies these operations whose nature and effect on database information are prescribed when privileges are defined. These operations could have the effect that they avail information with or without modifying the information in the database.

Traditional role-based security finds application in environments where the greater concern is information integrity as opposed to secrecy. Yet this does not preclude the exploitation of the advantages of role-based protection to realize secrecy. This paper presents a means of structuring role-based protection to enforce secrecy. Further, with additional rules on update and read operations, and the information they access, we can realize the requirements of *mandatory access control*, or MAC. It is our intention to demonstrate that a MAC-like level of protection can be realized using role-based security.

In MAC, access to objects by subjects is determined by subject and object attributes which form the sole basis for granting access. All accesses are arbitrated by a reference monitor (Department of Defense (1985)). In systems that realize MAC such as *multilevel security*, MLS, these subject and object attributes are partially ordered clearance and sensitivity levels, respectively. Moreover, the reference monitor applies two key rules: *no-read-up* and *no-write-down* rules whose effect is to ensure that information can flow from low to high sensitivity to high sensitivity levels (class-category pair) (Denning (1976)). The information flow relationship among these sensitivity levels is reflexive, transitive and antisymmetric (Denning (1976)) resulting in a lattice. In other words, with the sensitivity levels as nodes in a graph, the information flow relationship would be acyclic. A key strength of MAC is that it ensures information secrecy as well as offering safeguards against Trojan horse attacks. This is achieved by the requirement that information flows from low sensitivity levels to higher sensitivity levels and prevents information downgrade by assuring that information *cannot* flow from high to low sensitivity levels.

To emulate MAC, we apply information flow analysis (Nyanchama & Osborn (1994b), Nyanchama (1994)). In our role-based approach, we regard information accessed via a role as a *context*. Moreover, we require that all information flows among various contexts impose an acyclic information flow structure. To guard against cyclic flows, we impose rules that are the equivalent of the *no-read-up* and *no-write-down* rules, the simple security property and the *-property, respectively, of the Bell and LaPadula model (Bell & LaPadula (1975)). This will have the effect of imposing secrecy in the role-based system. This is in addition to the requirement of mediated access (ensuring the reference monitor principle) and system audit trail.

Traditional MAC provides for sensitivity levels (class-category pairs) which form a lattice (Denning (1976)). Every system object must be labeled with a sensitivity level. The term *class* will be used in this paper to refer to classes in an object-oriented environment. The term *context* will be used to refer to the information in a system accessible via a role. A role definition thus specifies a role's context.

Why are we interested in all this? Among the reasons is the observation that role-based protection offers key advantages in the management of access rights. Moreover, unlike traditional MAC which is based on simple reads and writes, privileges in role-

based protection systems can be complex. This gives us the benefit of complex operations while assuring system security. It allows for the exploitation of advantages of role-based protection while realizing MAC-like protection.

In the next section we present role and role-based protection basics. We define roles and discuss role-based security with an emphasis on its protection strategies. We define role contexts, discuss information flow in role-based protection and demonstrate the imposition of acyclic information flow in role-based schemes. Central to the concern for acyclic flows are the *read* and *update* scopes of a role.

Section 3 summarizes the basics of mandatory access control and proposes a means of enforcing MAC-like protection using roles. In particular, in section 3.1 we address the concepts such as subject and object attributes that form the basis of access, we discuss the means via which MAC tames Trojan horses. We show that central to this approach is the imposition of an acyclic information flow structure on a given system. Consequently, an acyclic information flow structure and subject-object attribute requirement in which a reference monitor manages all access will be deemed to realize mandatory access control. Section 3.2 combines the concepts of the two foregoing sections to propose a realization of mandatory access control in role-based protection. The key concerns here are (1) acyclic information flow among role contexts and (2) equivalent rules to the simple security and *–properties of traditional multilevel security. This is in addition to mediated access based on subject and object attributes.

The last section offers this paper's summary and conclusions.

2 ROLES & ROLE-BASED SECURITY

This section offers an overview of role-based protection. Roles are defined in terms of privileges. Hence, in discussing role basics in section 2.1 we define the term privilege as used in this paper. Role-based security is the subject of section 2.2 where we present an overview (a more comprehensive treatment can be found in Baldwin (1990),Thomsen (1991), Nyanchama & Osborn (1993a/b), Lawrence (1993), Nyanchama & Osborn (1994a) and Nyanchama (1994)) of the use of roles for information protection. An important notion in our formulation is the role context. Role contexts are important in that we treat them as the equivalent of security levels. This is the subject of section 2.3. The concept of information flow is useful for designing and analyzing a system's preservation of *secrecy*, an important requirement of traditional MAC. Hence section 2.4 address this matter starting with information flow basics.

2.1 Roles: Definition & Basics

A role can be seen as a job, a function, an encapsulation of rights, responsibilities and obligations, etc. in a system. It can be viewed as a specification of access rights at the disposal of a user authorized to the role. A role can be considered to be both *structural* and *functional* (Dobson & McDermid 1989), where a role's structure refers to its relationship with other roles in a system while a role's function pertains to that which can be done via executing the role. In our formulation, we define roles in terms of privileges where a privilege is a unit of access rights administration. Formally, a privilege is defined as:

Definition 1 *Privilege: A privilege is a pair (x, m) where x refers to an object (or object identifier) and m is a non-empty set of access modes for object x. It is unforgeable and its modification must be authorized.* □

A given privilege definition can be seen as a specification of a computation pertaining to object x and its access modes m. Depending on the set m (such as input and output parameters, other related invocations), a privilege can cause change (such as a change in object state) of, reveal or add to system information. It can cause the creation or deletion of objects. It can create new privileges or delete existing ones. It can create new roles or delete existing ones, etc. In general, the nature of a privilege is application-dependent. Consequently, our formulation offers a privilege definition that is of a general form and which can be specialized to a given application.

A privilege is defined in terms of an object's access modes. Each such mode of access has a particular effect on associated objects. For instance, executing some privilege can cause revelation of information pertaining to its parameters or it may cause modification of their associated information. Therefore, care must be taken to ensure that their definition achieves the desired effect when they are executed. For instance it may be the intention of a system designer to have a mode of access that reads information associated with its input objects and stores it in some output object. In this case we have information read from some objects and used to modify other objects. We may also have situations where we have the equivalent of simple reads that reveal particular information. The important issue is that these executions be executed accurately, according to specification.

Yet another important observation is that privileges can pertain to simple or complex objects. In accessing some object, an access mode need not access all the information associated with the object. Hence considering a given mode of access we are interested in that part of an object that the mode accesses. The net impact of executing a privilege is the cumulative effect of execution of all its access modes.

We designate the universal set of privileges in a system by $\mathcal{PV}$. Role definition follows from privilege definition. A role is a collection of privileges. Formally:

Definition 2 *Role: A role is a named collection of privileges. Its is a pair (rname,rpset) where rname is the role name and rpset is the privilege set.* □

For a given role **r**, **r.rname** and **r.rpset** refer to the name and privilege set of **r**, respectively. From a computational view, a role specifies the set of computations possible via authorization to the role. The effects of invoking all the role's computations would equal, *at least*, the union of the effects due to computations due to the individual privileges of the role's privilege set. We denote the universal set of roles in a system by $\mathcal{R}$.

2.2 Role-Based Security

Roles facilitate access to system resources and hence can be used to enforce security in a system. A role places at the disposal of an authorized user the resources accessible via the role. User-role authorization is one of the three forms of authorizations in role-based protection schemes (Baldwin (1990), Nyanchama & Osborn (1994a))*. In this form of authorization, a user/group is authorized access to system privileges available via the role. Such authorization must be specified in a role's access control list. For each role, the

*The others are role-privilege and role-role authorizations.

access control list contains the identifier for each user authorized to the role (Nyanchama & Osborn (1994a)). Formally:

Definition 3 Role Access Control List: *A role access control list (racl) is of the form:* $\{id_1, \cdots, id_n\}$ *where* $id_i \in ID$ *is a user* ($uid \in \mathcal{UID}$) *or group identifier* ($gid \in \mathcal{GID}$), *where* $\mathcal{ID} = \mathcal{UID} \cup \mathcal{GID}$. □

The access control list facilitates the determination of a user's authorization to a given role. In a secure system, no access, other than that specified in the role's access control list, is permitted. Consequently, in a secure system, each role must have an associated access control list, i.e. $\forall r \in \mathcal{R}, \exists r.racl = \{\cdots, id_i, \cdots\}$. This then leads to the concept of a *secure role* which we define as a role with an associated access control list used to determine authorization to the role.

Definition 4 Secure Role: *A secure role is a named collection of privileges along with its access control list. It is a triple (rname,rpset,racl), where rname is the role name, rpset is it s privilege set and racl is its access control list.* □

Determining whether there is authorized access for a given user to some object in some access mode is a two-stage process. First we ensure there is user-role authorization, i.e. the user's/group's identifier is in the role's access control list. Secondly, we ensure that the desired access mode to the object specified exists in the privilege set. The latter can be termed *role-privilege* authorization. This implies a two-stage process to confirm authorization: that the subject is authorized to a role and the role contains the associated privilege for access to the object. The latter is termed *object accessibility* via a role. The mode of access specified in the associated privilege is referred to as the legal mode of access via the role.

The process of determination of the roles to which a given user has authorization involves checking each role's access control list for the user's or user group's identifier. This has complexity $O(nm)$ with n being the number of roles in the system while m is the size of the access control list. In the worst case m is the number is users in the system. In cases of small access control lists, m is small and the complexity tends to $O(n)$.

Role execution rights are distinct from role administration rights, collectively termed access rights. Generally, no subject should have both types of access rights to any single role for that would result in conflict of interest. Administrative rights for a given role would usually be specified within a different role and authorization to these two roles must respect the conflict of interest principle.

2.3 Roles & Role Information Contexts

A privilege has been defined as a pair (x, m) with x being an object identif ier and m a set of *valid* access modes for x (definition 1). This definition prescribes that the privilege execution will be guaranteed to get its input from named sources (Glasgow, MacEwen & Panangaden (1992)), in this case the object x and any of the associated parameters of the access modes. This execution facilitates a *potential* transformation. It causes some information to be available via the execution and may cause information to flow in the process. Moreover, since the execution pertains to system information, it offers a form of access to system information. Each privilege can thus be seen as a *subcontext*

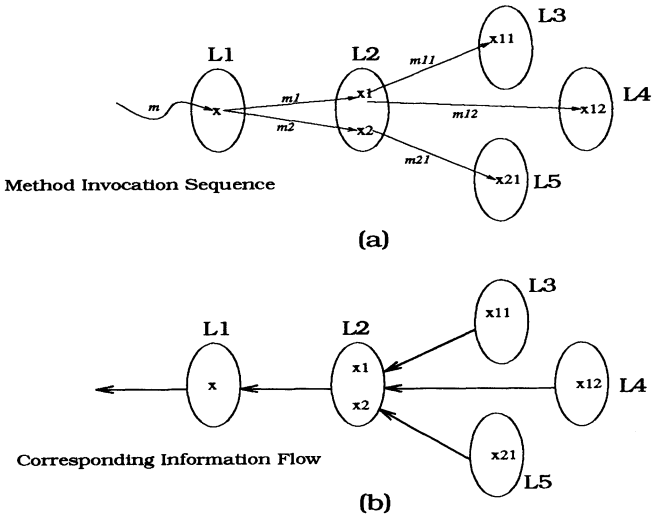

Figure 1 Method Invocation Information Flow

of information access. The choice of the term subcontext is deliberate since even when information is accessible via such a privilege, it must be accessed within some role.

Definition 5 *Privilege Information Subcontext: Privilege information subcontext is that piece of system information accessible via a given privilege.* □

To explain this further, we consider O-O methods as the basis of privil ege definition. Let x be the object identifier and m be a set of methods valid for x. The methods may have parameters other than x and their execution may invoke other methods or create new objects. In general, a method invocation sequence can be represented by a tree (figure 1a) (Jajodia & Kogan (1990)). In figure 1 the m_i's represent method invocations, the $C1, \cdots,$ are system-defined security levels (those involved in the invocation) while the x_i's stand for objects associated with a method invocation. The arrows in figure 1a indicate those invocations which cross security levels or objects, while those in figure 1b show the direction of information flow occasioned by the invocations.

Such invocations can facilitate access to information in the different security levels involved in the invocations. The information accessible from a security level via an invocation will be no greater than the information contained in the security level. An invocation such as that of figure 1 yields access to more than one security level and hence a fragment of information from each of the security levels in question. A subcontext of an invocation is the aggregate of all information fragments from the individual security levels involved in the invocation. In our case, the aggregate of information fragments accessed in security levels $C1, \cdots, C5$ constitute the subcontext of m. These fragments of information, in turn, are associated with the objects involved in the invocation. Therefore, the subcontext of m would contain the fragments of information pertaining to the x_i's in the figure.

The method in this case can be seen as a mini-window into system information pertaining to all the objects "touched" by the invocation. It is the information available via this

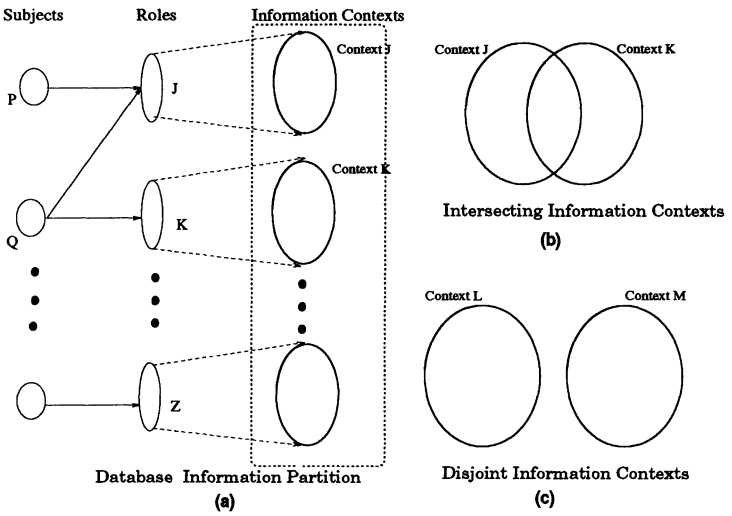

Figure 2 Information Partition Via Roles

mini-window that we call a subcontext for the given privilege. The associated *potential* information flow is depicted in figure 1b.†

A role facilitates access to a given set of objects using the specified modes of access in the associated privilege set. Seen this way a role acts as a *window* to system information. Information available via the role window is determined by the role's privilege set. It is, at the *least*, the union of all the information available via the individual privileges in its privilege set.

Given some role $r \in \mathcal{R}$ with an associated privilege set $r.rpset$, let $INF(t)$ represent the "quantity" (some measure) of information accessible via some role or privilege t. Where some pv is in some role r's privilege set, it is true that:

$$INF(pv) \leq INF(r)$$

In other words, the measure of information in a privilege cannot exceed that in its associated role. Yet, the "sum" of all information in a role's privileges is *always* less than or equal to the information available via the role. It follows that:

$$INF(r) \geq \bigcup_{pv \in r.rpset} INF(pv)$$

Notice that this is an inequality relationship, and not equality due to the principle of *aggregation* (Denning & Shockley (1992)): the total information of a "whole" is greater

†Note that we call this potential flows because we assume that all invocations cause some changes. Hence in the case that all such invocations associated with changes, we shall expect these flows. In practice, however, some invocations may not cause any changes and hence may not cause information flow (see axiom 1).

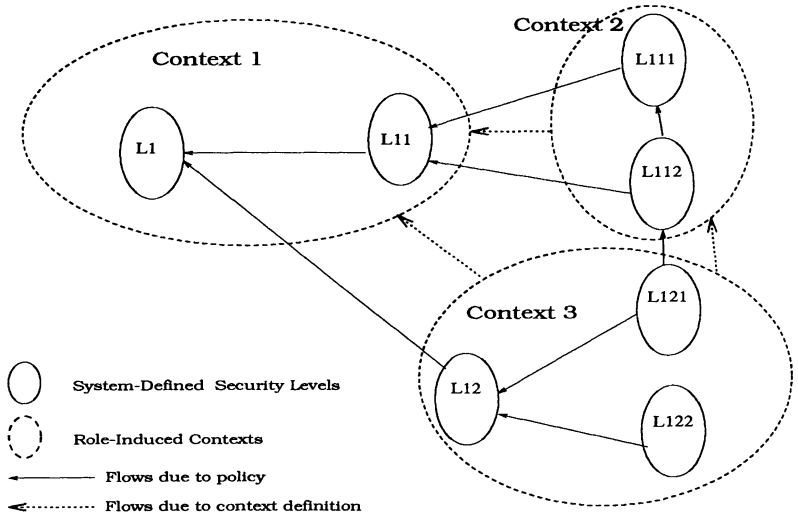

Figure 3 Policy & Context Information Flow

than or equal to the sum of the information from the individual parts that constitute the whole. For a given role r, its window of information is defined as $INF(r)$.

Consequently, a role-based system can be seen as *partitioning* system information and availing it via the windows defined by the roles, (see figure 2). The information available via one such window composes a context associated with the role. Each of these contexts is available to users via user authorization to the role. Formally:

Definition 6 *Role Information Context: A role's information context is that part of system information available via the role.* □

A distinction must be made between role-induced context (definition 6) and a system-defined security level. The former is associated with a particular role while the latter is a system-defined grouping of information within which information can flow freely.

Note that these system-defined security levels *need not* coincide with role contexts. To determine the context of a given role and its relationship with system-defined security levels, we take each privilege and determine its subcontext and how it straddles the system-defined ones (figure 3). In the most general case these subcontexts straddle more that one system-defined security level. It is possible that several security levels belong to one context. Moreover, several contexts can belong to one security level (see figure 4).‡

Different relationships can exist between contexts induced by a role definition scheme. These may or may not overlap. A context can be a proper subset of another context

‡Note that the different relationships shown in this figure constitute the most general cases. It is possible that any one of the shown cases may not satisfy the acyclic information flow requirement. The purpose of the figure is to demonstrate the scope of relationships that can exist between security levels and role contexts.

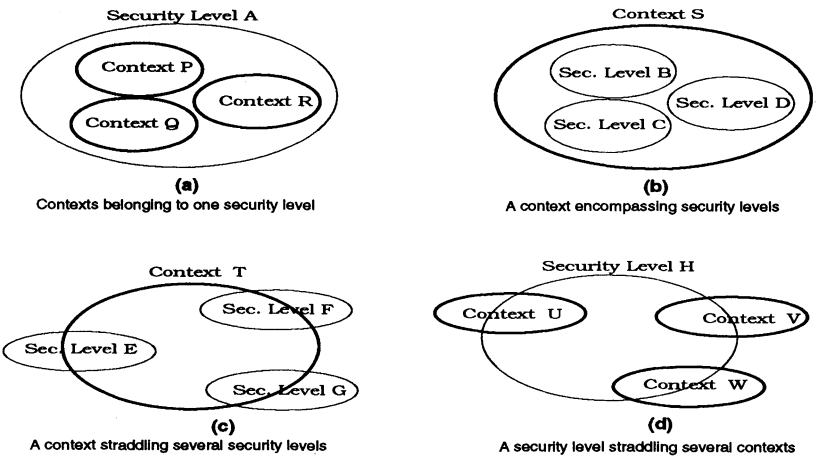

Figure 4 Security Level-Context Relationships

or indeed, it can be equivalent to another one when they are associated with roles with equivalent privilege sets.

2.4 Roles & Information Flow

In this section we offer a short summary (see a more detailed analysis in Nyanchama & Osborn (1994b)) of information flow analysis in role-based systems. This is important since we shall use the approach to ensure role-based security equivalence of mandatory access control. We start with the basic information flow problem proposed by McHugh (1985):

The information Flow Problem: Given a program and its sets of input and output variables, determine for each output variable, the subset of the input variable set about which it might contain information after execution of the program.

Information is said to flow from the subset of inputs to the outputs which contain information about this subset of inputs. Implicit in this statement is the fact that, for there to be an information flow, an input causes change in the output to which information flows. We have the following information flow axiom from Denning & Denning (1977) and Liu (1980):

Axiom 1 *Basic Information Flow axiom: A flow $x \longrightarrow y$ occurs only when the value of y is updated.* □

Information flow is transitive; hence the following transitivity axiom:

Axiom 2 *Information Flow Transitivity Axiom: Information flow is transitive, i.e. given that $X \longrightarrow Y \wedge Y \longrightarrow Z \Rightarrow X \longrightarrow Z$.* □

To analyze information flows from one security level to another, one must analyse the

operations involved. In particular we have the the basic information flow axiom (axiom 1) which holds that there is information flow when there are updates. Moreover, axiom 2 holds that information flow is transitive. Two kinds of operations can be found in systems: read operations which do not alter the state of the associated objects and update operations which alter/update/write states of the associated objects. An operation that does reads and updates can be seen as causing information to flow from the inputs that are read to the outputs that are updated.

Update operations within a role's context can be seen as defining the context's update scope. It is via these operations that information associated with a context is altered. Consequently, it is via such operations that update effects of the operations can be felt in other security levels. Moreover, it is via the same operations that information can flow from one context to another context. In a similar manner we can specify the read scopes of roles. A role's read scope facilitates access to information via the role without causing any side effects. We designate the read and update scopes by *r_scope* and *u_scope*, respectively. Hence for a role $r \in \mathcal{R}$, r_scope(r) and u_scope(r) refer to the read and update scopes of r, respectively.

From the basic information flow axiom we can make some general statements about information flows using the read and update scopes for a given role. Clearly, an operation that does both read and update would cause information to flow from the role's read scope to its write scope. Moreover, for an operation that does multiple updates, information flows from within the update. We have the following generalizations for a given role $r \in \mathcal{R}$ with r_scope(r) and u_scope(r), the read and update scopes, respectively:

1. $r\_scope(r) \longrightarrow u\_scope(r)$ for all $r \in \mathcal{R}$. This implies there is information flow within a role, i.e. $r \longrightarrow r$.
2. $u\_scope(r) \longrightarrow u\_scope(r)$ for all $r \in \mathcal{R}$. This kind of flow, like the one before is in agreement with the confinement problem.
3. In general, $r\_scope(r_i) \not\longrightarrow r\_scope(r_j)$ for $r_i, r_j \in \mathcal{R}$, i.e. there is no information flow across read scopes of two different roles (except that described in the next point).
4. Where there is overlapping of scopes of different roles we can expect information to flow between the two associated contexts. Consider two roles r_i and r_j with the following scopes: r_scope(r_i)={x,y,z}, u_scope(r_i)={p,q} for role r_i and r_scope(r_j)={a,b,c}, u_scope(r_j)={d,e,f}. From items 1 and 2 we have: {x,y,z}$\longrightarrow${p,q} and {a,b,c}$\longrightarrow$ {d,e,f} which are information flows due to r_i and r_j, respectively.
   Information flows from r_i to r_j, i.e. $r_i \longrightarrow r_j$ if and only if the updates due to r_i are accessible via r_j, i.e. either p or q or both belong to r_j's scope. In other words if we have $r_i \longrightarrow r_j$ then either u_scope(r_i) $\cap$ r_scope(r_j)$\neq \emptyset$ or u_scope(r_i) $\cap$ u_scope(r_j)$\neq \emptyset$.
   Multidirectional information flow occurs when we have both $r_i \longrightarrow r_j$ and $r_j \longrightarrow r_i$ designated $r_i \longleftrightarrow r_j$
   Unidirectional information flow can either be $r_i \longrightarrow r_j$ or $r_j \longrightarrow r_i$. Suppose it is required that we have $r_i \longrightarrow r_j$, it follows that:
   (u_scope(r_i) $\cap$ r_scope(r_j)$\neq \emptyset$ or u_scope(r_i) $\cap$ u_scope(r_j)$\neq \emptyset$) and
   (u_scope(r_j) $\cap$ r_scope(r_i)$= \emptyset$ or u_scope(r_j) $\cap$ u_scope(r_i)$= \emptyset$).

Definition 7 *Subsumed Information Flow: A subsumed information flow is one in which all flows from one context lead into exactly one context.* □

Suppose such a flow is of the form $r_i \longrightarrow r_j$. Then the conditions for such a flow are: either u_scope(r_i) $\subseteq$ u_scope (r_j) or r_scope(r_i) $\subseteq$ u_scope(r_j).

3 ROLES & MANDATORY ACCESS CONTROL

This section gives an overview of MAC and proposes a means of realizing MAC-like protection using roles.

3.1 MAC Basics

Access control is a means of protection that realizes information protection by limiting access of subjects to objects. In MAC, subjects must be given access rights in an explicit manner and meet the requirements for access before being allowed access. These access requirements are specified in terms of subject and object attributes. Hence authorization is determined on whether some subject's attributes meet the specification for access to objects of given attributes. Moreover, such security attributes are system-defined and can neither be changed by the subjects nor by the associated objects. Hence the specification and administration of access rights is a system security function that may be bestowed upon some super-user or a system security officer (Gasser (1988)).

In systems that implement MAC, subjects and objects are labeled and rules for access are specified. For instance in multilevel security, subject and object labels are the clearance and sensitivity levels. These sensitivity labels and clearance levels are the object and subject attributes upon which access is based. Access to an object by some subject depends on boththe simple security property and the *–property (Jensen, Kiel & Verjinski (1989), Gasser (1988)). Management of access is facilitated via the reference monitor through which all access must be channeled.

The major strength of mandatory access control is its ability to tame *Trojan horses.* A Trojan horse is hidden code within a program performing some legitimate function that uses the access rights of the subject running the program to violate the security policy (Jensen, Kiel & Verjinski (1989)), by masquerading as useful and in the process leaking information. A Trojan horse can release information in in two major ways: direct release or via indirect leakage (Denning (1985)). In the former, the Trojan horse behaves in a manner that causes information to be incorrectly labeled and hence released to otherwise authorized subjects. In the latter, a Trojan horse indirectly releases information by encoding it within authorized information which is returned to a subject, as in the case where it may appear that some query is being answered while in actual fact a different one is being answered (Denning (1985)).

The strategy employed in taming Trojan horses is based on the requirement for acyclic information flow. Hence a subject that reads information in one level must not write objects in another level which will violate the acyclicity principle. Hence a subject that reads secret information cannot write to objects that are either confidential or unclassified for that would imply downgrading of information. As well, a subject that writes information in a given level cannot read information in a level that might cause acyclic information flow.

3.2 Realizing MAC in Role-Based Security

This section will propose a scheme for realizing MAC in role-based protection systems. We start with the observation that in MAC information flow between levels is necessarily acyclic. Moreover, subjects are given clearances while objects are assigned sensitivity labels. Further, access is governed by the key rules: no-read-up and no-write-down. Combining all these and the requirement of a reference monitor, we crystallize the key tools for modeling MAC in role-based protection. Hence in this formulation, treat every role context as a security level. We then impose an acyclic information flow requirement in the way we structure roles. Further, we impose equivalent rules to those in MAC while we treat authorization to a role as a user's label. Access is based on this authorization as well as the role privileges being *bona fide* privileges for a given object.

To emulate mandatory access control in role-based protection systems, we must ensure that the system mandatory access function relies solely on the facts that a subject has *authorization to a role*, that the role contains an associated privilege specifying the mode of access to the object, that the subject access is via this legal mode, and that the access does not violate the specified flow policy (see constraint 1). Legal access of the object must be enforced, too, to ensure that the object is accessed in no modes other than those specified in the role as well as any other constraints specified on such access. Also, a flow policy must be observed since it is the criterion that determines security.

Constraint 1 *Information Consistency Constraint: A system is secure, with respect to some security policy $\mathcal{P}$, if the flows $\mathcal{F}(\mathcal{I})$ resulting from the system implementation are consistent with the flows, $\mathcal{F}(\mathcal{P})$, specified by the policy.* □

To ensure *secrecy* in our model, the information flow graphs of our role-based schemes must be acyclic. Where there are cycles, the set of all the roles/scopes in the cycle must necessarily be in the same context. To limit the effect of Trojan horse attacks, we must formulate an equivalent of the *-property to govern our role-based protection schemes. Moreover we determine what security attributes govern information authorization.

Further it must be specified what subject and object attributes would govern access of subjects to objects. Since a role facilitates subject access to objects or resources via the role, we can use this fact to specify MAC. In doing so we use two key subject/object attributes: *user authorization* to a role and *object accessibility* via a role.

Constraint 2 *Mandatory Authorization Constraint: A subject can only access an object via an authorized role in the mode specified in the associated privilege in the role.* □

Authorization to a role is specified in the role's access control list (see definition 3). Moreover, in a secure system, all roles are secure roles (see definition 4).

User authorization to a role means that the user can access objects accessible via the role via the specified (legal) modes of access to these objects. Users may be authorized access to more than one role. The authorization scheme must ensure that it does not violate the specified system security policy. Users can execute more than one role at any one time which does not violate the acyclic information flow imposed on the system. Where there is conflict between two or more roles, as where their execution may cause cyclic flows, the system will reject such executions. In its simplest form, this approach reduces to the state where a user executes only in a single role at any one time which is a stricter form of enforcement in that we need only ensure that the user is authorized to the

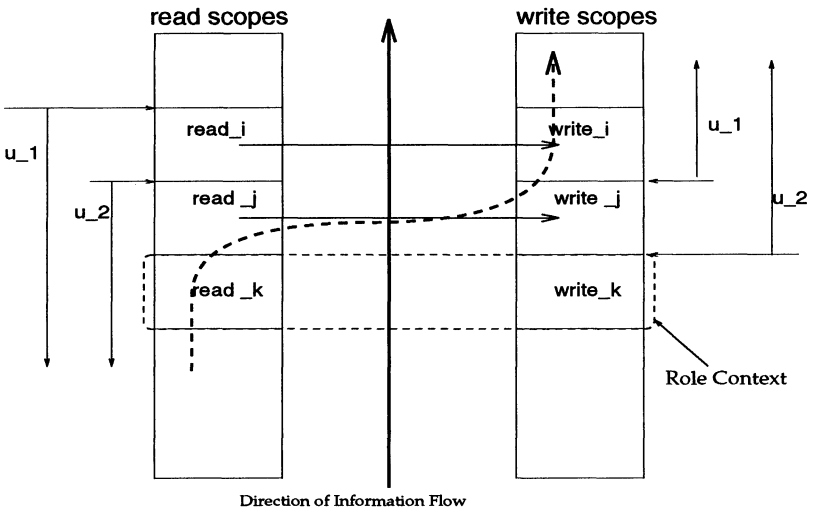

Figure 5 Unidirectional Information Flow in Roles

role; we need not care whether such execution would conflict with the acyclic information flow requirement. (In traditional MAC, no user logs onto the system in more than one level of clearance.) In order to execute different assigned roles where there is potential for conflict, a user must log out every time they need to change the roles in which they are executing.

The foregoing implies that a user's set of authorized roles can be divided into groups of roles which do not conflict, i.e. in any one such a group, an authorized user can freely execute any role without violating the acyclic flow of information. Hence at log on the user can be compelled to acquire execution rights of just one of these groups. Execution rules can be specified that ensure that the user executes only one group of roles at anyone time. Should it be desirable to execute in a role from another group, then the the user must log out and log on to acquire execution rights of the desirable group.

User authorization alone, however, is not sufficient to guarantee both secrecy and integrity of the information. It must be ensured that no such authorization will result in illegal information flow.

The rest of the constraints below, necessary to ensure MAC-like protection in roles, achieve the same effect that as that of Bell and LaPadula's *no read-up* and *no write-down* rules. Figure 5 offers the intended role organization that would ensure such protection.

Constraint 2 itself is not sufficient to guarantee secrecy. Indeed, while determination of authorization is a system function, we must ensure that secrecy cannot be violated due to overlapping scopes. Hence the following constraint:

Constraint 3 *Read (Secrecy) Access Constraint: Given two users, u_1 and u_2, and two roles, r_i and r_j, let u_1 have access to both read scopes and u_2 have access to the read scope of r_j. Then r_scope(r_j) must be a subset of r_scope(r_i), i.e. r_scope(r_j) $\subseteq$ r_scope(r_i).* □

Recall that information flows from a role's read scope into its update scope, i.e. r_scope(r_i)

$\longrightarrow$ *u_scope*(r_i) and *r_scope*(r_j) $\longrightarrow$ *u_scope*(r_j). Suppose that *r_scope*(r_j) $\not\subseteq$ *r_scope*(r_i). Then it means that there is information in *r_scope*(r_j) outside *r_scope*(r_i). But given that u_1 is authorized to both *r_scope*(r_i) and *r_scope*(r_j), we can have *r_scope*(r_i) $\longrightarrow$ *u_scope*(r_j). Hence if *r_scope*(r_j) $\not\subseteq$ *r_scope*(r_i) then there is information in *r_scope*(r_i) that is not guaranteed to flow into *u_scope*(r_i). In other words, u_2 has access about r_i that can be made to flow elsewhere, unless this information is a subset of r_i's read scope.

In specifying legal information flows and user authorizations, we must ensure that the read and update operations performed via different roles do not violate the specified flow policy. In other words, it should not be possible for a Trojan horse acting legally (via authorized writes) to leak information to an unauthorized context. The following two constraints are intended to guard against Trojan horse attacks:

Constraint 4 *Update Access Constraint: A subject cannot access one role's read scope and update another's update scope if there are no defined legal flows from the first role to the second.* □

The purpose of constraint 4 is to ensure that an information flow is defined in the direction of the update. This is due to the basic information flow axioms which say that information flows when there are updates.

Constraint 5 *Read/Update Constraint: A subject can access one role's read scope and update another's update scope if and only if the read scope of the second role contains the read scope of the first one.* □

In other words, given two roles r_i and r_j, subjects can write via r_i what other subjects in r_j can read if and only if there is defined a legal information flow (directly or indirectly) from the information context specified via *Context*(r_i) to that specified via *Context*(r_j).

Given that we have information of the form: $r_i \longrightarrow r_j$, the need for secrecy requires that r_j's read scope contain r_i's read scope.

From the foregoing, we conclude that MAC-like protection can be realized using role-based security if role definition and user-role authorization obey constraints 1, 2, 3, 4 and 5. These constraints ensure an implementation with respect to a given policy, they govern user-role authorization as well as the nature of access to information via the authorized roles.

Our result is similar to that of Thomsen (1991). While ours focused solely on general emulation of MAC in role-based protection by considering information flow, Thomsen's approach is with respect to well-formed transactions (WFTs) (Clark & Wilson (1987)) and the read and write sets of roles and their relationships. A role's write-set (read-set) is the context of objects (information) which can be written (read) by subjects authorized to the role. The proposed *Role-Based Security Property* states:

Property 1 *The Role-Based Security Property (RSP) of Thomsen (1991): Given two roles r_1 and r_2, subjects can write via r_1 what other subjects in r_2 can read if:*
1. a subject in r_2 can read any entity that r_1 can read
2. r_1 can only use WFTs to write entities readable by r_2, or
3. r_2 can only use WFTs to read entities written by r_1. □

In the absence of WFTs (see property 1), only the first item is useful here. This is a formulation we define based on the concept of *information flow.*

4 CONCLUSIONS

We have presented in this paper an emulation of mandatory access control using role-based protection. We made the key observation that in MAC we are interested both in the integrity and secrecy of information. Thus in MAC, information flows must be acyclic. We also observed that MAC requires subject and object attributes as the basis for granting authorization. Moreover, such authorizations must observe the reference monitor principle. Consequently, to realize MAC using role-based protection, we view each role context as a security level and ensure that information flows, caused either by role execution or user-role authorization will be acyclic. We proposed a number of access constraints that would realize the equivalent of Bell and LaPadula *no read-up* and *no write-down* rules. Moreover, user labels will be determined by authorization while subject labels are the bona fide privileges on the objects.

5 ACKNOWLEDGEMENTS

We acknowledge financial assistance of NSERC and of The University of Western Ontario through a Graduate Research Fellowship.

REFERENCES

R. W. Baldwin. Naming & Grouping Privileges to Simplify Security Management in Large Databases. In *Proc. 1990 Symposium on Res. in Security & Privacy*, pages 116–32. IEEE Computer Society Press, May 1990.

D. E. Bell and L. J. LaPadula. Secure Computer Systems: Unified Exposition & Multics Interpretation. Technical Report MTIS AD-A023588, MITRE Corporation, July 1975.

D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Security Policies. In *Proc. 1987 Symposium on Res. in Security & Privacy*, pages 184–94. IEEE Computer Society Press, April 1987.

D. E. Denning and P. J. Denning. Certification of Programs for Secure Information Flow. *Communications of the ACM*, 20(7):504–13, July 1977.

D. E. Denning. Commutative Filters for Reducing Inference Threats in Multilevel Database Systems. In *Proc. 1985 Symposium on Res. in Security & Privacy.* IEEE Computer Society Press, April 1985.

J. E. Dobson and J. A. McDermid. Security Models and Enterprise Models. In Landwehr, editor, *Database Security II: Status & Prospects*, pages 1–39. North-Holland, 1989.

D. E. Denning and W. Shockley. Discussion: Pros and Cons of the Various Approaches. In T. F. Lunt, editor, *Research Directions in Database Security*, pages 97–103. Springer-Verlag, 1992.

Morrie Gasser. *Building a Secure Computer System.* Van Nostrand Reinhold Company New York, 1988. ISBN 0-442-23022.

J. Glasgow, G. MacEwen, and P. Panangaden. A Logic for Reasoning About Security. *ACM Transactions on Computer Systems*, 10(3):226–64, August 1992.

S. Jajodia and B. Kogan. Integrating an Object-Oriented Data Model with Multilevel Security. In *Proc. 1990 Symposium on Res. in Security & Privacy*, pages 76–85. IEEE

Computer Society Press, May 1990.

C. D. Jensen, R. M. Kiel, and R. D. Verjinski. SDDM A Prototype of a Distributed Architecture for Database Security. In *Proc. of Int'l Conf. on Data Eng.*, pages 356–64, Feb 1989.

E. V. Krishnamurthy and A. McGuffin. On the Design & Administration of Secure Database Transactions. *ACM SIGSAC Review*, pages 63–70, Spring/Summer 1992.

L. G. Lawrence. The Role of Roles. *Computers & Security*, 12(1):15–21, Feb 1993.

L. Liu. On Secure Flow Analysis in Computer Systems. In *Proc. 1980 Symposium on Res. in Security & Privacy*, pages 22–33. IEEE Computer Society Press, April 1980.

J. McHugh. An Information Flow Tool for Gypsy. In *Proc. 1985 Symposium on Res. in Security & Privacy*, pages 46–48. IEEE Computer Society Press, April 1985.

M. Nyanchama and S. L. Osborn. Role-Based Security, Object Oriented Databases & Separation of Duty. *ACM SIGMOD RECORD*, 22(4):45–51, Dec 1993.

M. Nyanchama and S. L. Osborn. Role-Based Security: Pros, Cons & Some Research Directions. *ACM SIGSAC Review*, 2(2):11–17, June 1993.

M. Nyanchama and S. L. Osborn. Access Rights Administration in Role-Based Security Systems. In J. Biskup, M. Morgenstern, and C. Landwehr, editors, *Database Security VIII: Status & Prospects*, pages 37–56. North-Holland, August 1994.

M. Nyanchama and S. L. Osborn. Information Flow Analysis in Role-Based Security Systems. *"All about nothing", Journal of Computing & Information*, 1(1):1368–84, May 1994. Special Issue: Proc. of the 6th International Conference on Computing and Information (ICCI), Peterborough, Ontario, Canada.

Matunda Nyanchama. *Commercial Integrity, Roles & Object-Orientation.* PhD thesis, Department of Computer Science, The University of Western Ontario, London Ontario, N6A 5B7, Canada, September 1994.

Department of Defence. *Department of Defence Trusted Computer System Evaluation Criteria DoD 5200-28-STD.* Department of Defence, Dec 1985. The Orange Book.

T. C. Ting, S. A. Demurjian, and M. Y. Hu. Requirements Capabilities and Functionalities of User-Role Based Security for an Object-Oriented Design Model. In C. E. Landwehr and S. Jajodia, editors, *Database Security V: Status & Prospects*, pages 275–96. North-Holland, 1992.

D. J. Thomsen. Role-Based Application Design and Enforcement. In S. Jajodia and C. E. Landwehr, editors, *Database Security, IV: Status and Prospects*, pages 151–68. North-Holland, 1991.

6 BIOGRAPHY

Matunda Nyanchama received Msc and PhD degrees from the Department of Computer Science, University of Western Ontario in 1991 and 1994, respectively, and a Bsc (Electrical Engineering) from the University of Nairobi, Kenya, in 1981. His research interests include security, databases, distributed systems and networks.

Sylvia Osborn is an Associate Professor in the Computer Science Department at the University of Western Ontario, London, Canada. She received her PhD crom the University of Waterloo in 1978. Her research has been concerned with various aspects of databases, including normalization algorithms for relational databases, relational databases with complex domains, object-oriented databases and database security.