Information Flow Analysis of an RBAC System

Sylvia L. Osborn
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A-5B7
sylvia@csd.uwo.ca

ABSTRACT

Role-based access control provides a very flexible set of mechanisms for managing the access control of a complex system with many users, objects and applications. In our previous research, we have shown how, given a role graph and security labels for objects, one can test whether or not the system satisfies properties for lattice-based access control. In this paper we give a general mapping, which takes an arbitrary role graph and produces another graph which shows the information flow that can result from the roles defined in the role graph. An extension builds the information flow graph taking user assignments and sessions into account.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: [Access controls]; D.4.6 [**Security and Protection**]: [Information flow controls]

General Terms

algorithms, security

Keywords

role-based access control, role graphs, mandatory access control

1. INTRODUCTION

Role-based access control (RBAC) models have received a lot of attention in recent years. Two versions are the Sandhu model (we will refer here to RBAC96 [7, 6]) and the Nyanchama and Osborn role graph model [2]. There has also been research relating RBAC to more traditional discretionary access control (DAC) models and mandatory access control models (MAC), which can be found in [3]. MAC models are also referred to as lattice-based access control (LBAC), because the security levels of a MAC model are arranged in a lattice [5]. In [3], a mapping is given from an LBAC system to an RBAC system which allows and enforces the same access to data. Osborn, Sandhu and Munawer [3] also contains a section in which, given a role graph where each data item is labeled with a security label, we can test whether or not the system adheres to MAC principles. In this paper, we will attempt to give a more general mapping, where, given an arbitrary role graph where data is not labeled, we will construct another graph which shows the information flow implied by the role graph. An extension builds the information flow graph taking user assignments and sessions into account.

For any given RBAC system, it is trivially possible to construct an LBAC system in which there is one security level which is assigned to all data items. Our goal here is to try to show with as much detail as we can, both the information flow resulting from the RBAC system defined, and possibly a lattice of security labels which would achieve the same effect. We do not expect users to map RBAC systems to LBAC systems. Rather we envisage this work being used to provide information flow analysis to those users who wish to understand their system from that point of view. Since RBAC systems can be very large, it can be difficult to understand all the implications of one's RBAC design. This mapping into an information flow will provide one more tool to help security administrators understand their task and their systems.

The paper is organized as follows: the next section introduces the role graph model and the Sandhu model. This is followed by a brief summary of LBAC models in section 3. In section 4, we look at the information flows produced by the roles in a role graph. Section 5 extends this mapping by considering user-role assignments and constraints on sessions. The paper concludes in section 6 with a discussion of how a security lattice might be constructed from the flow graph.

2. THE ROLE GRAPH MODEL

The role graph model was introduced in [1]. A more complete description can be found in [2]. In the model, a role is defined as a named set of privileges, where a privilege is a pair (o, a), consisting of an object o in the system and one of its available access modes, denoted by a. The presence of (o, a) in the privilege set of a role indicates that users assigned to this role are allowed to perform this access on this object. In the role graph, roles are represented by the nodes of an acyclic directed graph. For each role r, there is a set Direct(r), the direct privileges of r, which are privileges (newly) available to r, and not in any of r's junior roles. Effective(r) denotes all privileges available to users assigned to r, and is the union of Direct(r) and all

SACMAT'02, June 3-4, 2002, Monterey, California, USA.

the privileges in any role junior to r. The edges in the role graph denote the is-junior relationship: $r_i \rightarrow r_j$ if and only if Effective(r_i) $\subset$ Effective(r_j). The role graphs correspond to what is called the role hierarchy in Sandhu's models. It is possible to introduce a MaxRole to which every role is-junior, and MinRole which is-junior to every role. If MaxRole is not already present, Effective(MaxRole) is just the union of all the privileges in the system, and if no MinRole is present, Effective(MinRole) = ϕ. Sandhu's examples do not contain MinRole and MaxRole; MaxRole is included in the role graph model to have some place to represent all possible privileges in a system.

A number of algorithms for manipulating role graphs are given in [2]: algorithms to insert and delete roles, insert and delete edges, and insert and delete privileges to/from a role. They all have run time which is polynomial in the size of the graph and in the cardinalities of the Effective and Direct sets. Role graphs are acyclic. A role graph is denoted by $\mathcal{G}(\mathcal{R}, \rightarrow)$ where $\mathcal{R}$ denotes the set of roles and $\rightarrow$ denotes the is-junior relationship.

The set of effective privileges of a role represents all privileges any subject assigned to that role can exercise. They correspond to the PA relationship of the RBAC96 model. The role graph model also includes a user plane and a privileges plane. The user plane can be used to model groups of users, which do not need to be considered in this paper. The privileges plane is meant to model implications among privileges (for example, in object-oriented databases, a very rich set of privilege implications can be defined [4]). We assume that all such privilege implications have been taken into account when defining the Direct and Effective sets for the roles.

In Sandhu's model, in addition to RH, the role hierarchy which corresponds to the role graph, and PA which corresponds the the assignment of privileges to roles in the role graph model, UA and Sessions are given. The UA contains all (user, role) pairs such that the user can perform this role. Sessions correspond to one user acting in possibly many of their assigned roles (according to UA). In addition to these sets, constraints can be specified which might,for example, limit the activation of two roles at the same time in the same session.

3. LBAC MODELS

In an LBAC system, all subjects and objects are labeled with a security label. These security labels are taken from a set of labels for which a lattice is defined [5]. As well as the labeling of all objects and all subjects, access in an LBAC system is governed by rules. Read access is defined by the simple security property:

Simple Security Property: subject s can read object o only if the label of $s \geq$ the label of o in the security lattice.

Write access is governed by either the liberal $\star$-property or the strict $\star$-property, which are defined as follows:

Liberal $\star$-property: subject s can write object o only if the label of $s \leq$ the label of o.

Strict $\star$-property: subject s can write object o only if the label of $s =$ the label of o.

These properties ensure that all information flow is from low security levels to high security levels.

A lattice differs from a directed graph in that, in a lattice, for every pair of nodes, a unique least upper bound is defined. Whereas every lattice can also be considered to be a directed acyclic graph, the opposite is not true. The existence of a unique least upper bound is important in LBAC systems to classify combinations of data objects. An acyclic directed graph can provide much useful information about information flow, even if it does not satisfy the additional requirements to be a lattice.

LBAC systems normally refer to subjects, whereas RBAC systems refer to users. The notion of a session in the Sandhu models corresponds to a subject in the LBAC sense, i.e. it corresponds to a process acting on behalf of a user. In the first discussion below, we will ignore users and subjects. Subsequently we will include them and will refer both to the UA and possible sessions of the Sandhu models.

4. THE FLOW MAPPING OF THE ROLE GRAPH

In constructing the mapping from a role graph to an information flow graph, we observe that we are constructing a "can flow" relationship from objects to other objects. Such flows can result from the existence of combinations of privileges in a role. Specifically, if the privileges (o_1, r) and (o_2, w) are in the same role, then any user acting in this role can cause information to flow from o_1 to o_2.

We should also note that each object has one security label in an LBAC system, so that although in the RBAC system, (o, r) and (o, w) might be in different roles, in some sense these two operations would be available at the same security level in any LBAC system (every object must have a single label). Therefore, no matter where (o_1, r) and (o_1, w) are, information flows from (o_1, w) to (o_1, r), and from (o_1, r) to (o_1, w).

Consider, then, an arbitrary role graph $\mathcal{G}(\mathcal{R}, \rightarrow)$. In what follows, we make the following assumptions:

1. $\mathcal{R}$ is finite
2. for each role r $\in \mathcal{R}$, Effective(r) is finite
3. the access modes contained in the pairs representing the privileges assigned to roles consist of only read and write, denoted by r and w respectively.

Given a role graph $\mathcal{G}(\mathcal{R}, \rightarrow)$, Algorithm 1, FlowStart in Figure 1 constructs an initial representation $\mathcal{F}_1$ of the "can flow" relationship. In the output, the nodes of result $\mathcal{F}_1$ are labeled by individual privileges. The algorithm starts by looking at all privileges in all roles, and creating a node in the output graph for this privilege. The nodes in the output are also labeled with the role name, to keep track of where the privileges come from. Note that this gives a finite number of nodes in $\mathcal{F}_1$. Then the algorithm puts edges into the flow graph to represent flows which result from (o_i, r) and (o_j, w) being in the same role. Finally it puts edges between all nodes in $\mathcal{F}_1$ which deal with the same object, to force them to be considered equal in the flow graph, i.e. it puts in edges between nodes containing (o_i, r) and (o_i,w).

Figure 2 shows a role graph in which there are three data items, a, b and c, and in which roles are labeled by their role

Algorithm 1: FlowStart

Inputs: $\mathcal{G}(\mathcal{R}, \rightarrow)$, a role graph, including for all r $\in$ R, Effective(r)
Output: $\mathcal{F}_1(\mathcal{N}, \rightarrow)$, a flow graph, where each node n $\in \mathcal{F}_1$ is labeled with a privilege

Method:
for each $r_k \in \mathcal{R}$
for each privilege (o, a) in Effective(r_k)
create a node n $\in \mathcal{N}$ and label it with the role name and the privilege
for every pair of privileges in Effective(r_k) of the form (o_i, r) (o_j, w)
insert an edge in $\mathcal{F}_1$ from the node whose label contains (r_k, o_i, r) to the node whose label contains (r_k, o_j, w)
for each object o
for each pair of nodes n_i and n_j whose labels contain the same object o,
insert edges $n_i \rightarrow n_j$ and $n_j \rightarrow n_i$

Figure 1: Algorithm FlowStart

name and set of effective privileges. The algorithm FlowStart run on the sample role graph in Figure 2 results in the initial flow graph shown in Figure 3. The node labels of Figure 3 also contain the role name where the privilege originated, to make it easier to follow how some of the edges originated.

Our claim is that the output of FlowStart contains all edges representing any potential flows represented by the roles themselves. We will see shortly that additional flows can result if two or more roles are active in the same session.

It is clear that the flow graph output from this first algorithm is not acyclic. The next step is to collapse cycles. This is accomplished by the algorithm CanFlow in Figure 5. The result of algorithm CanFlow on our previous example is the graph in Figure 4. At this point, we could remove references to reads and writes, and to the originating roles, but this information might still be useful to someone studying a large example. Essentially in this first example, object a flows to b and c, and b and c must be given the same security label because of the structure of role R3.

Consider a second example role graph in Figure 6. Here only 4 objects are shown in the privilege sets, represented by H, M1, M2 and L. This role graph gives the flow shown in Figure 7. Figure 6 is a role graph that can be used to simulate an LBAC system with the strict $\star$-property, having the four labels represented by the data items in Figure 6 [3].

Now consider the role graph in Figure 8. This is actually the role graph produced by Construction 1 in [3]. Here again only 4 objects are shown in the privilege sets, represented by H, M1, M2 and L. The flow graph resulting from this role graph has 4 isolated nodes, one each for data items L, M1, M2 and H (the only edges in $\mathcal{F}_1$ generated by FlowStart are between (H,r) and H,w), (M1,r) and (M1,w), etc.), because none of the roles contain both read and write operations. In Construction 1 in [3] is the constraint that each session has exactly two roles yR and yW where y $\in$ {H, M2, M1, L}. It is this combination of roles active at the same time which results in the information flow from L to M1, L to M2, etc. For this reason, we will now look at flows which result from UA and combinations of roles possible in sessions.

5. FLOW GENERATED BY USER-ROLE ASSIGNMENT AND SESSIONS

In order to get a realistic idea of the information flow that can result from the activation of multiple roles in a session, we must also look at the User-Role Assignment (UA) mapping of Sandhu's model and at any constraints on sessions. If there are no constraints expressed on sessions, then we must assume that a user can activate all of their roles in one session, and account for all the information flow that results from this. This does not introduce any new nodes in $\mathcal{F}_1$, only more edges. The step shown in the algorithm fragment in Figure 9 must be added to the end of Algorithm 1.

Now consider if we had added MaxRole to the first example in Figure 2. MaxRole will contain all the permissions, and would cause flows from objects b and c to a. If MaxRole is not in UA for any user, it can never be activated, so the flows it implies can never occur. One might also have other roles in a role graph which are not assigned to any user, but are part of a design strategy to bundle up collections of privileges to be inherited by other roles. These latter roles do have resulting information flows that appear in the roles which inherit the bundles of privileges. MaxRole on the other hand is a special case in that its privilege set is not inherited by any other role by construction of the role graph. By ignoring all roles not appearing in UA, we reduce the number of edges initially generated in $\mathcal{F}_1$. In order to take these cases into account, we now modify the first line of Algorithm 1 as follows:

for each r_k in $\mathcal{R}$ **such that** r_k **appears in UA**

6. DISCUSSION

The graph produced by the two algorithms gives the "can flow" relationship resulting from a given role graph, UA and constraints on sessions. This graph may or may not be a lattice. It happens that for our examples, the resulting graph has been a lattice, but this will not always happen. To test if the resulting graph is a lattice, we must determine whether, for every pair of nodes in $\mathcal{F}_2$, there exists a unique least upper bound. If there is no common upper bound, we can add a top node to the graph. If the upper bound is not unique, we can merge all of the least upper bounds for a given pair of nodes into a single node. It is possible to perform these

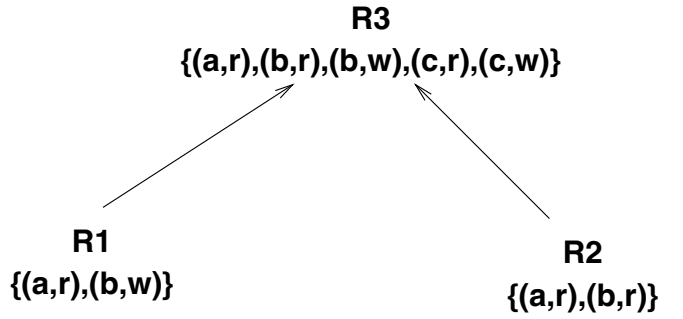

Figure 2: Example role graph

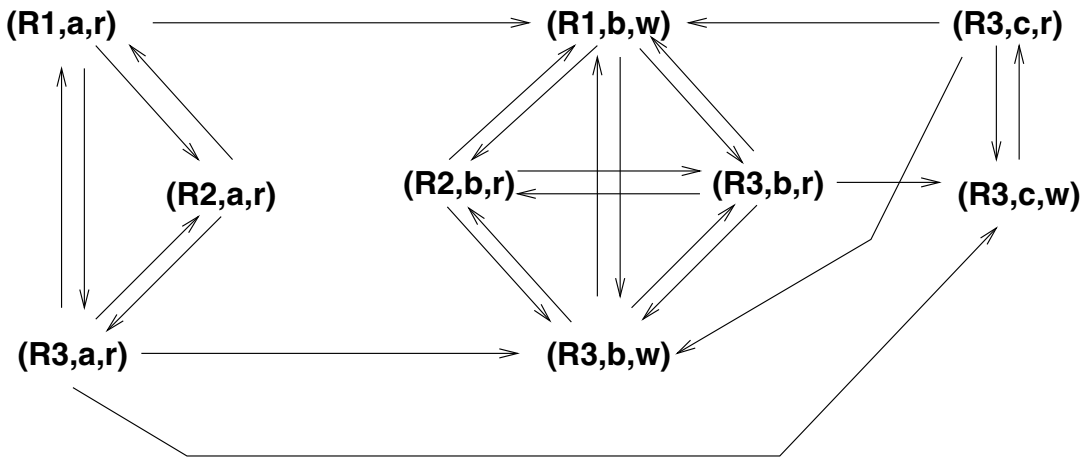

Figure 3: Resulting Initial Flow Graph

latter two operations in more than one way. More research is needed to see if there are efficient ways to carry out all of these operations. In a practical sense, if nodes in $\mathcal{F}_2$ are to be merged, a security designer may want to fine tune the choices. (And a default option is just to consider one security label for all the data, so trivially a solution always exists.) In any case, $\mathcal{F}_2$ is finite, and the operations being proposed do not change that, so that if $\mathcal{F}_2$ does satisfy the lattice properties, it does provide a candidate security labeling for the data objects in the original role graph.

Our discussion and algorithms in this paper, then, do not quite provide a clear mapping from an RBAC system to an equivalent LBAC system. We have, however, shown how to compute the resulting information flows, and discuss how to generate an equivalent lattice of security labels.

7. ACKNOWLEDGEMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

[1] M. Nyanchama and S. L. Osborn. Access rights administration in role-based security systems. In J. Biskup, M. Morgenstern, and C. E. Landwehr, editors, *Database Security, VIII, Status and Prospects WG11.3 Working Conference on Database Security*, pages 37–56. North-Holland, 1994.

[2] M. Nyanchama and S. L. Osborn. The role graph model and conflict of interest. *ACM TISSEC*, 2(1):3–33, 1999.

[3] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Information and System Security*, 3(2):1–23, 2000.

[4] F. Rabitti, D. Woelk, and W. Kim. A model of authorization for object-oriented and semantic databases. In *Proceedings of the International Conference on Extending Database Technology*, Venice, Italy, Mar. 1988.

[5] R. Sandhu. Lattice-based access control models. *IEEE Computer*, 26:9–19, Nov. 1993.

[6] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Trans. on Information and Systems Security*, 2(1):105–135, Feb. 1999.

[7] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29:38–47, Feb. 1996.

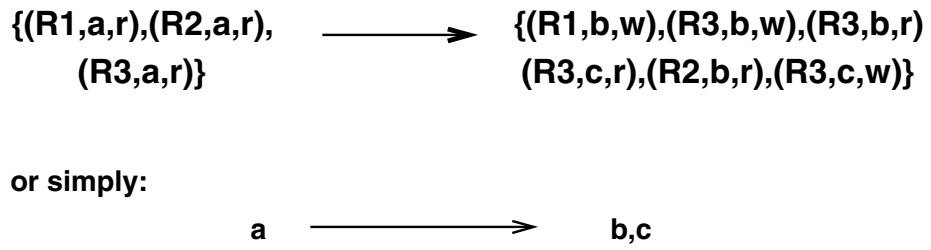

Figure 4: "Can Flow" Graph for the first example

Algorithm 2: CanFlow

Inputs: $\mathcal{F}_1(\mathcal{N}, \rightarrow)$, a flow graph possibly including cycles
Output: $\mathcal{F}_2(\mathcal{N}, \rightarrow)$, an acyclic flow graph where nodes are labeled with sets of privileges

Method:
copy $\mathcal{F}_1$ into $\mathcal{F}_2$
for every cycle in $\mathcal{F}_2$
replace all the nodes in the cycle by a single node whose label is the union of the labels in the cycle

Figure 5: Algorithm CanFlow

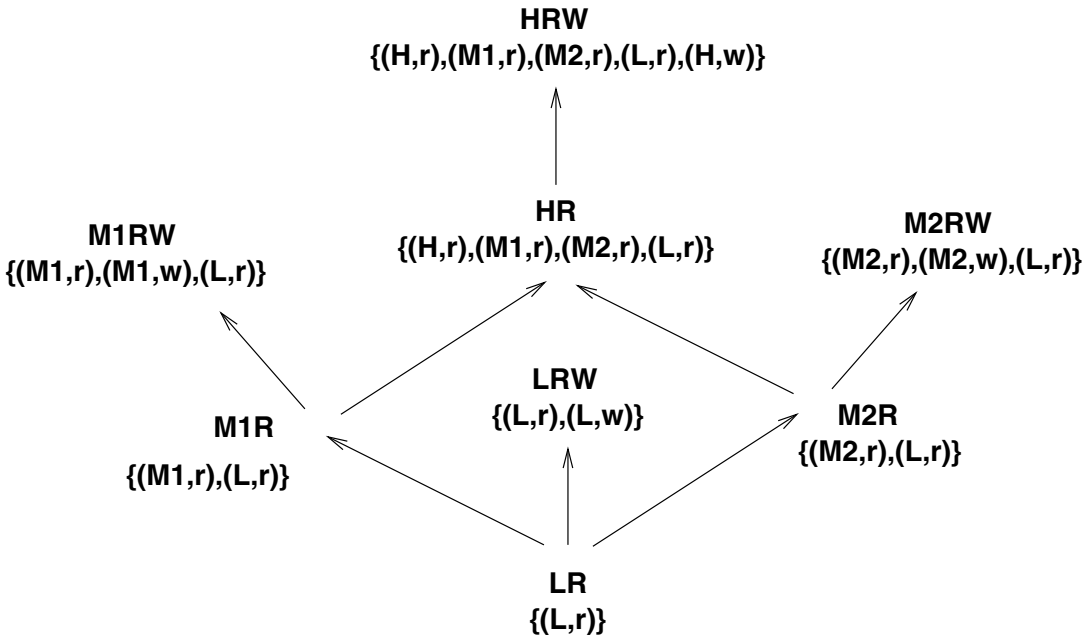

Figure 6: Example 2

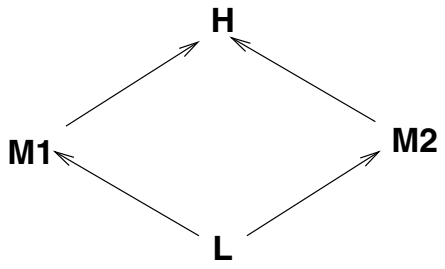

Figure 7: Flow Graph for Example 2

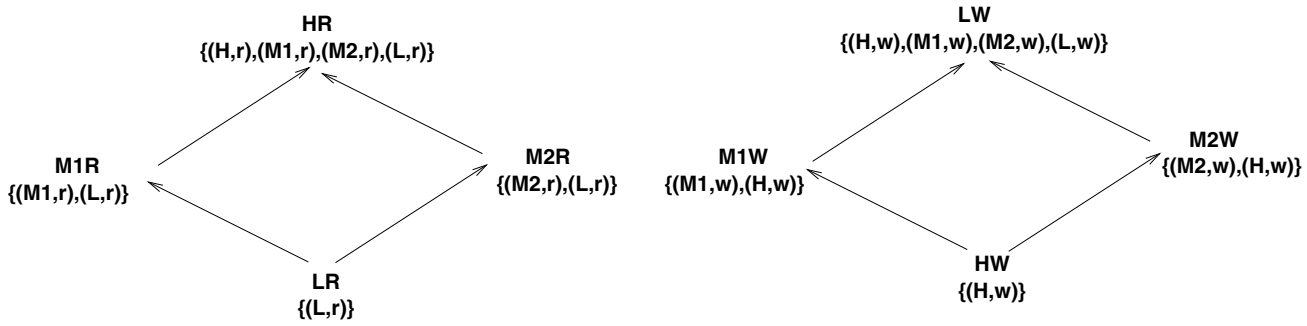

Figure 8: Example 3

for all roles r_i and r_j in UA for a single user, and permitted by constraints to appear in the same session for a single user
    for all pairs (o_k, r) in r_i and (o_m, w) in r_j
        add an edge from (r_i, o_k, r) to (r_j, o_m, w)

Figure 9: Algorithm Fragment to Take UA and constraints into consideration