JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 22, 49-61 (2006)

An Approach to XML-Based Administration and Secure Information Flow Analysis on an Object Oriented Role-Based Access Control Model

CUNGANG YANG AND CHANG N. ZHANG*
Department of Electrical and Computer Engineering
Ryerson University
Toronto, Ontario, M5B 2K3, Canada
**Department of Computer Science*
University of Regina
Regina, Saskatchewan, S4S 0A2, Canada

In this paper, a practical method that can be employed to manage security policies using the eXtensible Markup Language (XML) is presented. The method efficiently administrates security policies based on the object oriented role-based access control model (ORBAC). Moreover, an information flow analysis technique is introduced for checking whether or not a created XML-based ORBAC security policy satisfies the Mandatory Access Control (MAC) security principles.

Keywords: information flow, object oriented role-based access control model (ORBAC), XML, MAC, confinement problem

1. INTRODUCTION

Nowadays, there are three basic access control techniques: mandatory access control (MAC), discretionary access control (DAC) and role-based access control (RBAC) [1-3]. The central notion of RBAC is that users do not directly get access to enterprise objects; instead, access privileges for the objects are associated with roles; each user is assigned one or more appropriate roles. As a result, an organization cant not only consistently preserve an access control policy appropriate to its characteristics, but also maintains access control relationships between users and objects independently.

In [4-6], we proposed a variation of the RBAC model called the Object Oriented Role-based Access Control model (ORBAC). Though ORBAC is a good model, administration of ORBAC, including creating and maintaining an access control security policy, remains a challenging problem. Administrative RBAC (ARBAC) [7] provides an alternative solution that reduces the complexity of administration. However, it does not solve the fundamental problems of security policy administration.

Realizing the need for an enterprise environment that is flexible and manageable, researchers have proposed several frameworks, such as the OMG Resource Authorization Decision (RAD) specification [8], CORBA Security Service [9], and Secure Euro-

Received November 11, 2002; revised April 25, 2005; accepted August 31, 2005.
Communicated by Randy Y. C. Chow.

pean System in a Multi-vendor Environment (SESAME) [10]. Other papers, such as [11-13], have provided notation, logics, and calculi for expressing and reasoning about security policies. But these works separated security logics from application logics or mainly focused on modelling policies and enforcement mechanisms, and put little emphasis on managing security policies.

In this paper, a novel approach to facilitating security policy management is proposed. Our research employs XML-based technology for defining and representing security policies. The advantages of using XML are that as a meta language, it can well define ORBAC security policies and is able to extend and modify security policies easily. XML can precisely and effectively represent desired security policies and offer an additional degree of flexibility. In addition, XML allows a security manager to build and administrate a security policy for an entire enterprise. Since there may be a containment problem with the ORBAC model, an information flow analysis technique is introduced here. It checks whether or not the created XML-based ORBAC security policy satisfies the MAC security principles.

The rest of the paper is organized as follows. Section 2 presents an information flow analysis technique based on ORBAC. Section 3 proposes an XML-based approach for creating and administrating an ORBAC security policy. Finally, section 4 concludes the paper.

2. INFORMATION FLOW ANALYSIS BASED ON THE OBJECT ORIENTED ROLE-BASED ACCESS CONTROL MODEL

In the proposed ORBAC model, roles are divided into two groups: position roles and task roles. A position role is a collection of task roles performed by someone in a certain position in an enterprise. On the other hand, each task role can be assigned to one or multiple position roles. The position role hierarchy reflects the inheritance of authority and responsibility among position roles. A task role may have multiple privileges, and the same privilege can be associated with different task roles. We define objects and their object methods as the *privileges* of the model, and object methods are split into two different categories: *basic object methods* and *complex object methods*. If an object method is a function of an individual object and does not invoke methods of other objects, we call it as a *basic object method.* If an object method is not only a function of an object but also invokes object methods of other objects, we call it a *complex object method*. A user can be assigned a number of position roles, and a position role also can be assigned to multiple users. The formal class definition for each element of the model was discussed in [4], and a diagram which describes the relationships among position roles, task roles, complex object methods, and basic object methods in ORBAC is shown in Fig. 1.

In [14], we pointed out that there may be a *confinement problem* with ORBAC; that is, unauthorized users may access objects because of unsafe information flow. We define that there exists an information flow from object O_i to object O_j if and only if the information is read from object O_i and written to object O_j. Information flow from object O_i to object O_j is safe only if $\lambda(O_i) \leq \lambda(O_j)$, where λ signifies the security label of the indicated objects and users. Therefore, if $\lambda(O_i) > \lambda(O_j)$, then the information flow from object O_i to object O_j is unsafe. A user s can access object *o* if he/she satisfies two MAC security

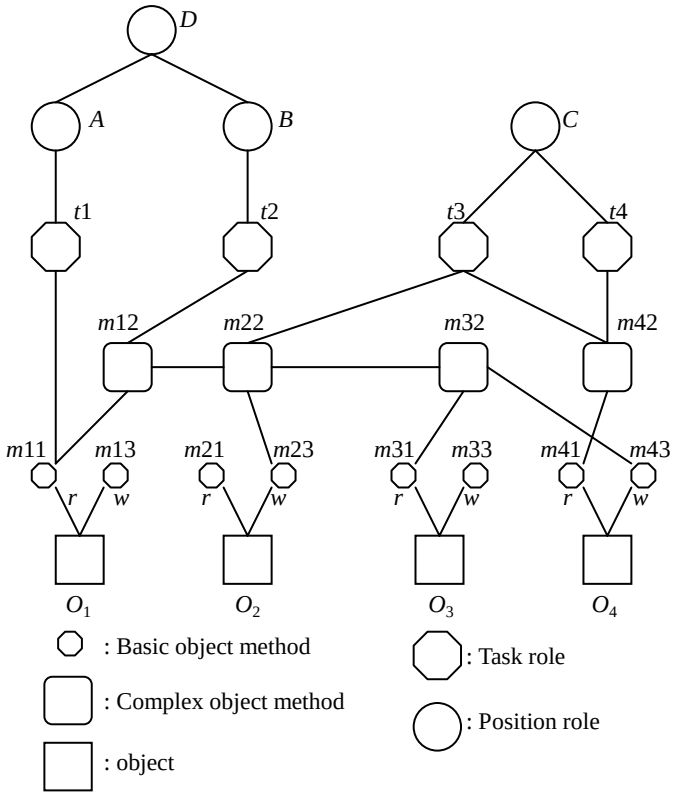

Fig. 1. Relationships among position roles, task roles, complex object methods, and basic object methods in ORBAC.

principles: the *Simple-Security Property* (user s can read object o only if $\lambda(s) \geq \lambda(o)$) and the $*$*–Property* (user s can write object o only if $\lambda(s) \leq \lambda(o)$). Moreover, we have proposed a *message filter* to automatically check whether there exists unsafe information flows for each task role in ORBAC [14]. The message filter intercepts every message exchanged between the objects in each task role and determines whether or not there are unsafe information flows.

Based on the message filter, we solve the confinement problem by proposing *a role set assignment method*. The steps in the role set assignment method are described as follows. (1) Assign a security label to each object and user; each object O_i in the model is classified by means of a security label, $\lambda(O_i)$, and each user s is classified by means of a security label, $\lambda(s)$. (2) Apply an information flow analysis technique to check information flows for each task role in ORBAC and every information flow within task roles is safe. (3) According to each position role's responsibilities, assign task roles to position roles; senior position roles inherit the task roles of their junior position roles. (4) Calculate a security label for each position role, and calculate a security label for each *position role set*; a position role set is a set of position roles which is assigned to a user by the security manager. We define that the r-scope of a position role R includes all the objects

upon which position role R has performed read operations, and the security label of a position role R, $\lambda(R)$, will be greater than or equal to the labels of each object (O_i, O_j, …, O_k) in position role R's r-scope. $\lambda(R) \geq (\lambda(O_i), \lambda(O_j), \ldots, \lambda(O_k))$. For instance, as shown in Fig. 1, assume $\lambda(O_1) = 1$, $\lambda(O_2) = 1$, $\lambda(O_3) = 2$, and $\lambda(O_4) = 2$; then, λ(role A) = 1, λ(role B) = 2, λ(role C) = 2, and λ(role D) = 2. Moreover, the security label of a position role set, S (position role set L), is greater than or equal to the label of all the position roles within the set {position role L_1, position role L_2, position role L_3, … position role L_n}. For instance, the security label of a position role set such as {A, C} should be greater than or equal to 2. (5) Assign position role sets to users; when a position role set L is assigned to a user T, it should be confirmed that the security label of the user, $\lambda(T)$, $\lambda(T) \geq \lambda(L)$.

3. IMPLEMENTING THE ORBAC SECURITY POLICY USING XML TECHNOLOGY

3.1 XML-Based ORBAC Security Policy

Our research employs XML for syntactic representation of security policies. The XML specification [15] is the work of the World Wide Web Consortium (W3C) Standard Generalized Markup Language (SGML) Working Group. Being a meta-language, it provides accessible notations and a means to describe an ORBAC conceptual model. Each ORBAC component is represented by an XML element, and an XML-based ORBAC security policy is comprised of the two parts described below.

Part 1: Basic elements of the XML-based ORBAC security policy.

- User is represented by

```
<! --User set definition-- >
<USER ID = user-id    USER LABEL = user-label></USER>
```

The above syntax defines a new XML tag of type USER with a required ID attribute value user-id and USER LABEL attribute value user-label.

- Object is represented by

```
<! --Object set definition-- >
<OBJECT ID = objecte-id    OBJECT LABEL = object-label></OBJECT>
```

The above syntax defines a new XML tag of type OBJECT with a required ID attribute value objece-id and OBJECT LABEL attribute value object-label.

- Privilege is represented by:

```
<! --Privilege set definition-- >
<PRIVILEGE ID = privilege-id    OBJECT = object-id    METHOD = m1>
</OBJECT>
```

The above syntax defines a new XML tag of type PRIVILEGE with a required ID attribute value privilege-id, OBJECT attribute value object-id, and METHOD attribute *m*1 (basic object method or complex object method) on object-id.

• Position Role is represented by

```
<! --Position Role definition-- >
<POSITION ROLE ID = role-id></POSITION ROLE>
```

The above syntax defines a new XML tag of type POSITION ROLE with a required ID attribute value role-id.

ORBAC places constraints on user-position role authorization. An example of a position role constraint is that a position role may have a limited number of members. For instance, there is only one person in the position role of chairman of a department; similarly, the number of position roles that an individual user can possess may also be limited. We call these constraints cardinality constraints.

The concept of prerequisite roles is based on the ideas of competency and appropriateness, whereby a user can be assigned to position role *A* only if the user is already a member of position role *B*. For example, only those users who are already members of a project role can be assigned to the testing task role within that project. In this example, the prerequisite position role is inferior to the new position role being assumed.

Separation of duty is an important theme in ORBAC, that is, the separation of duty (SOD) or conflict of interest constraints. The goal of SOD is to improve security by preventing collisions between two or more users. Some task roles cannot be assigned to the same user; otherwise, one user would be sufficient to perform some illegal activities. A well-known example of SOD involves the separate task roles involved when initiating a payment and authorizing a payment. In this case, no single position role that is assigned to a user should be capable of executing both task roles. If a task role is defined as having SOD constraints from another task role, then we call them conflicted task roles. No position role in a domain position role hierarchy can directly or indirectly inherit conflicted task roles. Moreover, there exist many other position role constraints (i.e., time constraints), which affect how long position roles can be activated.

• Constraint is represented by

```
<! --Constraint set definition-- >
<CONSTRAINT ID = constraint-id></CONSTRAINT>
    <PARAMETER VALUE = parameter-values></PARAMETER>
```

The above syntax defines a new XML tag of type CONSTRAINT with a required ID attribute of value constraint-id; the parameter values for the constraint are defined by means of a PAREMETER tag with a required VALUE attribute value parameter-value. PARAMETER VALUE varies for different kinds of constraints.

- Task Role is represented by:

<! --Task Role definition-- >
<TASK ROLE ID = task role-id></TASK ROLE>

The above syntax defines a new XML tag of type TASK ROLE with a required ID attribute value task role-id.

Part 2: The relationships between different elements.

Part 2 defines the relationships between different elements of the ORBAC model.

- The position role hierarchy is represented as a set of INHERITES elements, each of which associates a position role with its direct child position role. For instance, a position role hierarchy is represented by

<! --Role hierarchy definition-- >
<INHERITES FROM = *ri* To *rj*></INHERITES>

The above syntax defines a new XML tag of type INHERITES with required FROM (position role *ri*) and TO (position role *rj*) attribute values, which indicate that position role *ri* is a direct parent position role of position role *rj*.

- A privilege assignment assigns privilege (according to the object and basic object method or complex object method) to a task role, and it is represented by

<! --Privilege assignment definition-- >
<PRIV-ASSIGN TASK ROLE = *ti* PRIVILEGE = *mi*></PRIV-ASSIGN>

The above syntax defines a new XML tag of type PRIV-ASSIGN with TASK ROLE and PRIVILEGE attributes, in which task role *ti* has a privilege *mi* (according to the object and basic or complex object method).

- A task role assignment assigns a set of task roles to a position role, and it is represented by

<! --Task role assignment definition-- >
<TASK ROLE-ASSIGN POSITION ROLE = *ri* TASK ROLE = *tj*, …, *tk*>
</TASK ROLE-ASSIGN>

The above syntax defines a new XML tag of type TASK-ASSIGN with POSITION ROLE and TASK ROLE attributes, in which position role *ri* has task roles *tj*, …, *tk*.

- A position role assignment assigns a set of position roles to a user, and it is represented by

<! --Position role assignment definition-- >

<POSITION ROLE-ASSIGN USER = *u*1 POSITION ROLE = *r*1, …, *rm*>
</POSITION ROLE-ASSIGN>

The above syntax defines a new XML tag of type POSITION ROLE-ASSIGN with a required USER and POSITION ROLE attribute, in which user *u*1 is assigned a position role set *r*1, …, *rm* by the security manager .

- A constraint assignment assigns a set of position role constraints to a position role, and it is represented by

<! --Role constraint assignment definition-- >
<CONS-ASSIGN ROLE = *r*1 CONSTRAINTS = *c*1, …, *cm*>
</CONS-ASSIGN>

The above syntax defines a new XML tag of type CONS-ASSIGN with a required ROLE and CONSTRAINTS attributes, in which role *r*1 has constraints *c*1, …, *cm*.

- An example of a separation of duty constraint is represented by

<! --SSOD assignment definition-- >
<SSOD-CONSTRAINT SOD = *ri*, *rj*></SSOD-CONSTRAINT>

The above syntax defines a new XML tag of type SSOD-CONSTRAINT with required SOD attributes, in which task role *ri* and task *rj* are conflicted task roles.

Based on the definitions of the elements of the XML based ORBAC security policy, an example of a security policy is shown below.

Example 1:

```
<? xml version = “1.0”>
<ORBAC-MODEL TYPE = “RBAC1_POLICY”>
  <! --Basic Elements-- >
    <! --User set definition-- >
      <USER ID = “U1” USER LABEL = “2”></USER>
      <USER ID = “U2” USER LABEL = “2”></USER>
    </ --User set definition-- >
    <! --Object set definition-- >
      <OBJECT ID = “O1” OBJECT LABEL = “1”></OBJECT>
      <OBJECT ID = “O2” OBJECT LABEL = “1”></OBJECT>
      <OBJECT ID = “O3” OBJECT LABEL = “2”></OBJECT>
      <OBJECT ID = “O4” OBJECT LABEL = “2”></OBJECT>
    </ --Object set definition-- >
    <! --Privilege set definition-- >
      <PRIVILEGE ID = “p1” OBJECT = “O1” METHOD = “m11”></OBJECT>
      <PRIVILEGE ID = “p2” OBJECT = “O1” METHOD = “m12”></OBJECT>
      <PRIVILEGE ID = “p3” OBJECT = “O1” METHOD = “m13”></OBJECT>
      <PRIVILEGE ID = “p4” OBJECT = “O2” METHOD = “m21”></OBJECT>
```

```
        <PRIVILEGE ID = "p5"   OBJECT = "O2"   METHOD = "m22"></OBJECT>
        <PRIVILEGE ID = "p6"   OBJECT = "O2"   METHOD = "m23"></OBJECT>
        <PRIVILEGE ID = "p7"   OBJECT = "O3"   METHOD = "m31"></OBJECT>
        <PRIVILEGE ID = "p8"   OBJECT = "O3"   METHOD = "m32"></OBJECT>
        <PRIVILEGE ID = "p9"   OBJECT = "O3"   METHOD = "m33"></OBJECT>
        <PRIVILEGE ID = "p10"  OBJECT = "O4"   METHOD = "m41"></OBJECT>
        <PRIVILEGE ID = "p11"  OBJECT = "O4"   METHOD = "m42"></OBJECT>
        <PRIVILEGE ID = "p12"  OBJECT = "O4"   METHOD = "m43"></OBJECT>
    </ --Privilege set definition-- >
    <! --Position Role definition-- >
        <POSITION ROLE ID = "A"></POSITION ROLE>
        <POSITION ROLE ID = "B"></POSITION ROLE>
        <POSITION ROLE ID = "C"></POSITION ROLE>
        <POSITION ROLE ID = "D"></POSITION ROLE>
    </ --Position Role definition-- >
    <! --Constraint set definition-- >
        <CONSTRAINT ID = "C1"></CONSTRAINT>
            <PARAMETER VALUE = "h1h2"></PARAMETER>
        <CONSTRAINT ID = "C2"></CONSTRAINT>
            <PARAMETER VALUE = "h3"></PARAMETER>
        <CONSTRAINT ID = "C3"></CONSTRAINT>
            <PARAMETER VALUE = "h4h5"></PARAMETER>
        <CONSTRAINT ID = "C4"></CONSTRAINT>
            <PARAMETER VALUE = "h6"></PARAMETER>
        <CONSTRAINT ID = "C5"></CONSTRAINT>
            <PARAMETER VALUE = "h7h8"></PARAMETER>
    </ --Constraint set definition-- >
    <! --Task Role definition-- >
        <TASK ROLE ID = "t1"></TASK ROLE>
        <TASK ROLE ID = "t2"></TASK ROLE>
        <TASK ROLE ID = "t3"></TASK ROLE>
        <TASK ROLE ID = "t4"></TASK ROLE>
    </ --Task Role definition-- >
</ --Basic Elements-- >
<! --Relationships of Elements-- >
    <! --Role hierarchy definition-- >
        <INHERITES FROM = "D" To "A"></INHERITES>
        <INHERITES FROM = "D" To "B"></INHERITES>
    </ --Role hierarchy definition-- >
    <! --Privilege assignment definition-- >
        <PRIV-ASSIGN TASK ROLE = "t1"   PRIVILEGE = "p1"></PRIV-ASSIGN>
        <PRIV-ASSIGN TASK ROLE = "t2"   PRIVILEGE = "p2"></PRIV-ASSIGN>
        <PRIV-ASSIGN TASK ROLE = "t3"   PRIVILEGE = "p5"></PRIV-ASSIGN>
        <PRIV-ASSIGN TASK ROLE = "t3"   PRIVILEGE = "p11"></PRIV-ASSIGN>
        <PRIV-ASSIGN TASK ROLE = "t4"   PRIVILEGE = "p11"></PRIV-ASSIGN>
    </ --Privilege assignment definition-- >
```

```
<! --Constraint assignment definition-- >
    <CONS-ASSIGN ROLE = "A"  CONSTRAINTS = "C1"></CONS-ASSIGN>
    <CONS-ASSIGN ROLE = "B"  CONSTRAINTS = "C2"></CONS-ASSIGN>
    <CONS-ASSIGN ROLE = "C"  CONSTRAINTS = "C3, C5"></CONS-ASSIGN>
    <CONS-ASSIGN ROLE = "D"  CONSTRAINTS = "C4"></CONS-ASSIGN>
</ --Constraint assignment definition-- >
<! --Task Role assignment definition-- >
    <TASK ROLE-ASSIGN POSITION ROLE = "A"  TASK ROLES = "t1">
    </TASK ROLE-ASSIGN>
    <TASK ROLE-ASSIGN POSITION ROLE = "B"  TASK ROLES = "t2">
    </TASK ROLE-ASSIGN>
    <TASK ROLE-ASSIGN POSITION ROLE = "C"  TASK ROLES = "t3, t4">
    </TASK ROLE-ASSIGN>
</ --Task Role assignment definition-- >
<! --Position Role assignment definition-- >
    <POSITION ROLE-ASSIGN USER = "U1"  POSITION ROLE = "D">
    </POSITION ROLE-ASSIGN>
    <POSITION ROLE-ASSIGN USER = "U2"  POSITION ROLES = "B, C">
    </POSITION ROLE-ASSIGN>
</ --Position Role assignment definition-- >
</ --Relationships of Elements-- >
</ORBAC-model>
```

3.2 Administration of the XML-Based ORBAC Security Policy

The object model of an XML-based ORBAC security policy is automatically created by an object model translator. An example of a created object model of the XML-based ORBAC security policy in example 1 is shown in Fig. 2, and the general procedure for creating the object model shown in Fig. 3 is described below.

Step 1: The object model translator parses each line in part 1 of the security policy in example 1, creates objects of each element (user, privilege, position role, task role, and position role constraint) according to their class definitions as described in [4], and calls the function of the privilege objects to assign the values of *privilege-id* and *object label* fetched from the security policy to each created privilege object. In the same way, the object model translator assigns *position role-id* to each position role object. Also, position role constraints can be enforced by assigning *constraint-id* on each position role constraint object and by assigning *parameter-values* fetched from the security policy to the position role constraint object.

Step 2: The object model translator parses each line in part 2 of the security policy in example 1 and establishes the relationships between position roles and position roles, position roles and task roles, task roles and privileges, position roles and position role constraints, and users and position roles. An object model of the XML-based ORBAC domain security policy in sample was created and is shown in Fig. 2.

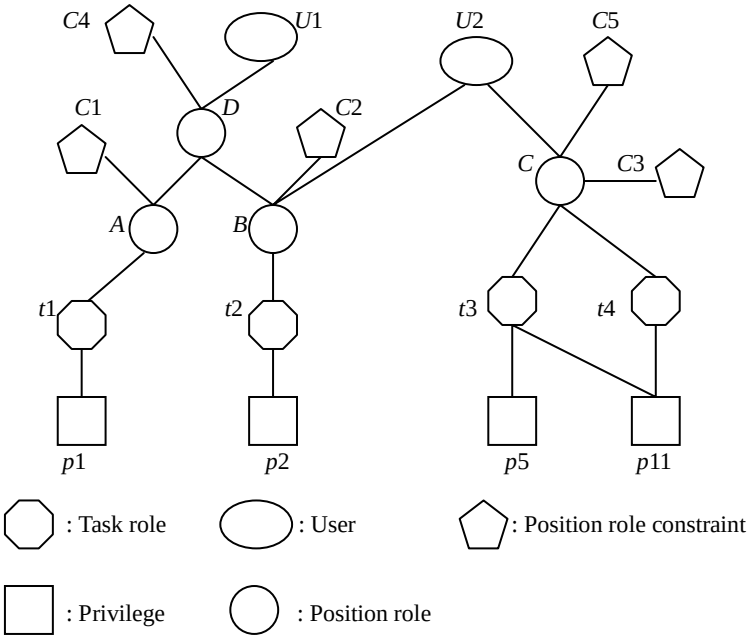

Fig. 2. An example object model of the XML-based ORBAC security policy in example 1.

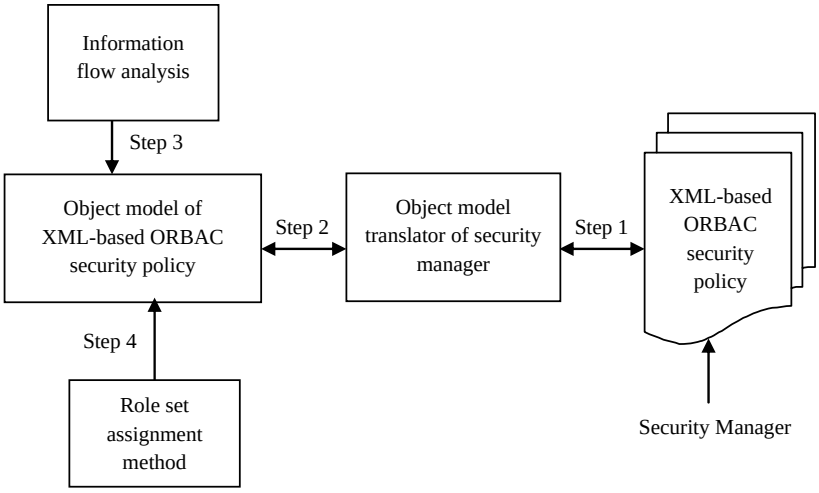

Fig. 3. Diagram showing the creation of an object model by security manager.

Step 3: Check the information flows in each task role of the security policy using a message filter and confirm that each information flow is safe. If one is not, the label of the objects should be modified or the security policy should be modified. After each modification, apply the message filter again and make sure each information flow is safe.

Step 4: Calculate the label of each position role using on the role set assignment method introduced in section 2. Calculate the label of each position role set and assign the position role set to users based on their security labels.

3.3 Modifying XML-Based ORBAC Security Policy

In an enterprise environment, the numbers of users, position roles, task roles, position role constraints, and privileges can be very large. Maintaining the secure status of a security policy is a formidable task when the basic elements of ORBAC change. The proposed XML-based approach to ORBAC security policies simplifies the required maintenance. If the security manager adds, deletes, or modifies some basic elements in an XML-based security policy, then the object model translator will automatically recreate a new object model, and the information flow analysis technique will be used to check and make sure that all information flows within task roles following the modification in the object model are safe.

4. CONCLUSIONS

In this paper, an XML-based security policy and information flow analysis technique has been presented. The motivation for this research is the need to simplify security policy administration for object-oriented RBAC systems. The main contribution of this paper is a new approach that can be used to efficiently manage XML-based security policies. Unlike most of the existing implementations, with our approach, authorization is independently defined and is separated from policy representation and from implementation mechanisms. We believe that our proposed approach can be applied to develop a generalized security policy language for expressing any necessary security policy for an enterprise environment.

REFERENCES

1. R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role based access control models," *IEEE Computer*, Vol. 29, 1996, pp. 38-47.
2. J. Crampton, "Specifying and enforcing constraints in role-based access control," in *Proceedings of 8th ACM Symposium on Access Control Models and Technologies* (*SACMAT 2003*), 2003, pp. 43-50.
3. N. Li, Z. Bizri, and M. V. Tripunitara, "On mutually-exclusive roles and separation of duty," in *Proceedings of 11th ACM Conference on Computer and Communications Security*, 2004, pp. 42-51.
4. C. N. Zhang and C. Yang, "Towards complete model of role-based access control system for distributed networks," *Journal of Information Science and Engineering*, Vol. 18, 2002, pp. 871-889.
5. C. N. Zhang and C. Yang, "Specification and enforcement of object-oriented RBAC model," *2001 IEEE Canadian Conference on Electrical and Computer Engineering*, 2001, pp. 65-77.

6. C. N. Zhang and C. Yang, "An object-oriented RBAC model for distributed system," in *Proceedings of the Working IEEE/IFIP Conference on Software Architecture* (*WICSA*), 2001, pp. 24-32.
7. R. S. Sandhu, V. Bhamidipati, E. Coyne, S. Ganta, and C. Youman, "The ARBAC97 model for role-based administration of roles: preliminary description and outline," in *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, 1997, pp. 155-162.
8. OMG CORBAmed DTF, Resourse Access Decision (RAD), revised submission, 1999.
9. OMG CORBA services common Object Services specification: CORBA Security Service v1.2, 1998.
10. P. Ashley and B. Broom, "An implementation of the SESAME security architecture for Linux," in *Proceedings of Australian UNIX and Open Systems Group Technical Conference*, 1997.
11. Y. Bai and V. Varadharajan, "A logic for state transformations in authorization policies," in *Proceedings of the IEEE Computer Security Foundation Workshop*, 1997, pp. 173-182.
12. S. JaJordia, P. Samarati, and V. S. Subrahmanian, "A logical language for expressing authorizations," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1997, pp. 31-42.
13. T. Y. C. Woo and S. S. Lam, "Authorizations in distributed systems: a formal approach," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1992, pp. 33-50.
14. C. N. Zhang and C. Yang, "Information flow analysis on role-based access control model," *Information Management and Computer Security*, Vol. 10, 2002, pp. 225-236.
15. Extensible Markup Language (XML) 1.0-W3C Recommendation 10-Feb-98, http://www.w3.org/TR/1998/REC-xml-19980210.

Cungang Yang received the M.S. degree in Computer Science from Jilin University, China. He completed his Ph.D. degree in Computer Science in 2003 at University of Regina, Canada. In 2003, he joined the Ryerson University as an assistant professor in the Dept. of Electrical and Computer Engineering. His research areas include wireless sensor network security, enhanced role-based access control model, information flow control, and multimedia security.

Chang N. Zhang received the B.S. degree in Applied Math from Shanghai University, China, and the Ph.D. degree in Computer Science and Engineering from Southern Methodist University. In 1988, he joined Concordia University as a research Assistant Professor in Dept. of Computer Science. Since 1990, he has been with University of Regina, Canada, in Dept. of Computer Science. Currently he is a full professor and leads a research group in parallel processing, data security and neural networks.