A Dynamic Mandatory Access Control Model*

Jafar Haadi Jafarian, Morteza Amini, and Rasool Jalili

Computer Engineering Department
Sharif University of Technology
Tehran, Iran
{jafarian@ce.,m_amini@ce.,jalili@}sharif.edu

Abstract. Mandatory access control has traditionally been employed as a robust security mechanism in critical environments like military ones. As computing technology becomes more pervasive and mobile services are deployed, applications will need flexible access control mechanisms. Aggregating mandatory models with context-awareness would provide us with essential means to define dynamic policies needed in critical environments. In this paper, we introduce a dynamic context-aware mandatory access control model which enables us to specify dynamic confidentiality and integrity policies using contextual constraints.

Keywords: Mandatory Access Control, Dynamicity, Context-Awareness, Confidentiality, Integrity.

1 Introduction

Mandatory access control is a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. Mandatory access control has traditionally been employed as a robust security mechanism in critical environments like military ones. Due to high heterogeneity and dynamicity, new computing environments such as pervasive environments require more flexible authorization policies which are mainly dependant on context; e.g. an operation can only take place in a specified time interval [1]. Aggregating mandatory models with context-awareness would provide us with essential means to define dynamic policies needed in new pervasive critical environments. In other words, incorporating context-awareness into mandatory models would enormously enhance their expressiveness as well as dynamicity.

In this respect, Ray et al. [2] proposed a location-based mandatory access control model which extends Bell-LaPadula model with the notion of location. In particular, every location is associated with a confidentiality level and Bell-LaPadula no read-up and no write-down properties are extended by taking confidentiality levels of locations into consideration. Nonetheless, context-awareness has never been applied to mandatory access control models. In other words, although location is considered as a

* This research is partially supported by Iran Telecommunication Research Center (ITRC).

H. Sarbazi-Azad et al. (Eds.): CSICC 2008, CCIS 6, pp. 862–866, 2008.

fundamental contextual value, Ray et al.'s model can not be considered as a context-aware mandatory access control model.

In this paper, we introduce a dynamic context-aware mandatory access control model which firstly preserves the confidentiality and integrity of information and secondly enables us to define dynamic access control policies required in pervasive critical environments.

2 A Dynamic Mandatory Access Control Model

The model we present in this paper is a dynamic context-aware mandatory access control model which is preserving both confidentiality and integrity while utilizing contextual information to enhance expressiveness and dynamicity of traditional mandatory models. In particular, a subject's access to an object can be contingent on contextual values as well as confidentiality and integrity axioms.

The model can be formally defined as a ten-tuple:

$\langle EntitySet, RepOf, ConfLvl, IntegLvl, \lambda, \omega, ContextPredicateSet, ContextSet, OperationSet, OperatorDefiner \rangle$

in which *EntitySet* is the set of all entities in the system and is composed of four sets: *UserSet, SubjectSet*, *ObjectSet* and *EnvironmentSet*. *UserSet, SubjectSet* and *ObjectSet* include all users, subjects, and objects of the system respectively. *EnvironmentSet* has only one member called *environment*.

- *RepOf: SubjectSet* $\rightarrow$ *UserSet* determines for each subject the user on behalf of whom the subject is acting.
- *ConfLvl* is an ordered set of confidentiality levels. For instance in Bell-LaPadula confidentiality model [3], *ConfLvl* can be defined as $\langle TS, S, C, U \rangle$.
- *IntegLvl* is an ordered set of integrity levels. For instance in Biba integrity model [4], *IntegLvl* can be defined as $\langle C, VI, I \rangle$.
- $\lambda : (UserSet \cup SubjectSet \cup ObjectSet) \rightarrow ConfLvl$ is a mapping function which associates each user, subject, and object, with a confidentiality level.
- $\omega : (UserSet \cup SubjectSet \cup ObjectSet) \rightarrow IntegLvl$ is a mapping function which associates each user, subject, and object, with an integrity level.
- *ContextPredicateSet* is the set of current context predicates in the system (described in section 2.1). *ContextSet* is a set of context types (introduced in section 2.2). *OperationSet* is the set of all operations in the system (defined in section 2.3).
- *OperatorDefiner* function defines operators in $OperatorSet = \{\leq, \geq, <, >, \neq, \subseteq, \supseteq, \subset, \supset, \not\subset, =\}$ on contextual values as well as confidentiality/integrity levels. In other words, *OperatorDefiner* determines that for two arbitrary values *A* and *B* whether *A op B* is true or not. The *OperatorDefiner* is added to the system as an external module.

2.1 Context Predicate

Each *context predicate* is a predicate which represents a value for a contextual attribute. We define a context predicate $cp \in ContextPredicateSet$ as a 4-tuple:

$$cp = \langle en, ct, r, v \rangle$$

in which $En \in EntitySet$, $ct \in ContextSet$, $r \in RelatorSet_{ct}$ and $v \in ValueSet_{ct}$. For example, $\langle John, Location, Is, Classroom \rangle$ is a context predicate which gives information about the current location of subject *John*.

If $\langle e, x, r, v \rangle$ is a context predicate, $x[e][r]$ indicates the value assigned to entity *e* for context type *x* and relator *r*. In other words, $x[e][r] = v$. For instance if $\langle John, Location, Is, Classroom \rangle \in ContextPredicateSet$, then $Location[John][Is] = Classroom$. If such a context predicate does not exist in *ContextPredicateSet*, we assume that $x[e][r] = \perp$ (read as null).

2.2 Context Type

Informally, *context type* can be defined as a property related to every entity or a subset of existing entities in the system. In fact, context type represents a contextual attribute of the system; e.g. time or location of entities. Formally, a context type like $ct \in ContextSet$ can be defined as a binary tuple:

$$ct = \langle ValueSet_{ct}, RelatorSet_{ct} \rangle$$

More detail on each component of *ct* is given below.

Set of Admissible Values: $ValueSet_{ct}$

$ValueSet_{ct}$ denotes the set of values that can be assigned to variables of context type *ct*. Set representation can be used to determine members of $ValueSet_{ct}$. For instance, the value set of context type *Time* can be defined using set comprehension as follows:

$$ValueSet_{Time} = \{n \in N \mid 0 \le n \le 24\}$$

Set of Admissible Relators: $RelatorSet_{ct}$

$RelatorSet_{ct}$ represents the set of admissible relators for context type *ct*. For instance, for context type *Location*, $RelatorSet_{Location}$ can be defined as follows:

$$RelatorSet_{Location} = \{Is, Entering, Leaving\}$$

2.3 Operations

OperationSet is the set of operations defined in the system. Formally an operation $opr \in OperationSet$ is defined as $opr = \langle AccessRightSet_{opr}, Constraint_{opr} \rangle$.

More details on each component of the Operation *opr* are given below.

$AccessRightSet_{opr}$

The set of access rights in our model is comprised of *read (r)* and *write (w)*. In this model, every operation, based on what it carries out, includes a subset of these access rights; e.g. if it only does an observation of information and no alteration, it only includes *r*, and so on. $AccessRightSet_{opr}$ is a subset of the set {*r, w*} which denotes access rights of the operation.

$Constraint_{opr}$

Each operation owns a *constraint* which denotes the prerequisite conditions that must be satisfied before the operation is executed. For, $opr \in OperationSet$ this constraint is represented by $Constraint_{opr}$.

Use of variables *USR*, *SBJ* and *OBJ* allows us to define an operation constraint in a general way. When a subject as a representative of a user, wishes to execute an operation on an object, the variables are replaced by their current values and then the constraint is evaluated.

Operation constraints are logical expressions built of *condition blocks (CB)*. Each *CB* declares a conditional statement that can be evaluated using *OperatorDefiner* function. Formally a condition block is a triple $\langle Value_1, op, Value_2 \rangle$ in which $Value_1, Value_2 \in (ValueSet \cup IntegLvl \cup ConfLvl)$ and $op \in OperatorSet$ and is evaluated using the *OperatorDefiner* function in the following way:

$$OperatorDefiner(Value_1, op, Value_2)$$

For example, $\langle Location[SBJ][Is] = University \rangle$ denotes that the location of subject must be a subset of the university.

Operation constraint is constructed using the following unambiguous grammar:

$$Constraint \rightarrow Constraint \vee C1 \,|\, C1$$

$$C1 \rightarrow C1 \wedge C2 \,|\, C2$$

$$C2 \rightarrow (Constraint) \,|\, CB$$

Definition of operations finalizes the specification of our model. Next, we clarify the process through which the users' requests are authorized.

2.4 Authorization of Users' Requests

A subject's request to access an object is represented by an *action.* Formally, an action A is a triple $\langle s, o, opr \rangle$ in which $s \in Subject`$, $o \in Object$ and $opr \in Operation$. Since each subject is a representative of a user, the user of an action is determined by $RepOf(s)$.

Before an action is granted, its constraint must be evaluated. The constraint of an action such as *act* is denoted by $Constraint_{act}$ and is initially equal to the constraint of its operation; i.e. for an action $act = \langle s, o, opr \rangle$, initially $Constraint_{act} = Constraint_{opr}$.

The access rights set of an operation determines which mandatory policies must be considered before the operation is executed. In order to preserve confidentiality, we use Bell-LaPadula policy. Also, to preserve integrity, Biba strict integrity policy is being used in the model. In particular, for an action $act = \langle s, o, opr \rangle$, if $r \in AccessRightSet_{opr}$, Bell-LaPadula Simple-Security property and Biba Simple-Integrity property must be added to the constraint of that action. In addition, if $w \in AccessRightSet_{opr}$, Bell-LaPadula *-property and Biba Integrity *-property must be incorporated into it.

$$if\ r \in AccessRightSet_{opr} \quad Constraint_{act} = Constraint_{act} \wedge (\langle \lambda(SBJ), \geq, \lambda(OBJ) \rangle \wedge \langle \omega(OBJ), \geq, \omega(SBJ) \rangle)$$

$$if\ w \in AccessRightSet_{opr} \quad Constraint_{act} = Constraint_{act} \wedge (\langle \lambda(OBJ), \geq, \lambda(SBJ) \rangle \wedge \langle \omega(SBJ), \geq, \omega(OBJ) \rangle)$$

Before the constraint of an action is evaluated *USR*, *SBJ* and *OBJ* are replaced by their current values. For example for an action $act = \langle s, o, opr \rangle$ and $u = RepOf(s)$, *USR*, *SBJ* and *OBJ* are replaced with *u*, *s* and *o* respectively.

After the replacement is done the constraint is evaluated using the *OperatorDefiner* function and if the result of evaluation is true, the action will be granted.

3 Evaluation and Conclusions

Our model could be evaluated and compared with other mandatory models based on the following criteria: complexity of policy specification, support for context-awareness, expressiveness and security objective. Due to the space limitation we only consider complexity and expressiveness in this paper.

In respect of complexity, although specification of access control policies in our model is dynamic and flexible, it is more complicated in comparison with Bell-LaPadula and other traditional MAC models. However, the dynamicity which it offers, justifies its complexity in policy specification.

Furthermore, various mandatory policies can be specified by our model. Models such as Dion [5] and policies like Chinese Wall [6] can be easily expressed using our model. Therefore, the model is highly expressive.

In this paper, we explained the need for a dynamic mandatory access control model and presented a model which satisfies such a requirement. Our model utilizes context-awareness to enable specification of sophisticated and dynamic mandatory policies. In addition, various mandatory controls can be incorporated into our model. In this model, Bell-LaPadula, and Biba strict integrity policy are designated as built-in, and Chinese Wall and Dion policies can be appended to the model using context types.

One of the main advantages of the model besides its dynamicity is the ability of deploying different combinations of MAC policies simultaneously in a system. For instance, Bell-LaPadula, Biba, and Chinese Wall Policies can all be deployed at once.

References

1. Ray, Kumar, M.: Towards a location-based mandatory access control model. Computers & Security 25, 36–44 (2006)
2. Masoumzadeh, R., Amini, M., Jalili, R.: Context-Aware Provisional Access Control. In: Proceedings of the Second International Conference On Information Systems Security, Kolkata, India (2006)
3. Bell, D.E., LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation. MITRE Corporation (1976)
4. Biba, K.: Integrity Considerations for Secure Computer Systems, Bedford, MA (1977)
5. Dion, L.C.: A Complete Protection Model. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA (1981)
6. Sandhu, R.S.: A Lattice Interpretation of the Chinese Wall Policy. In: Proceedings of the 15th NIST-NCSC Nat'l Computer Security Conference, Washington, D.C (1992)