Modeling Context for Access Control Systems

G. Sladić*, B. Milosavljević* and Z. Konjović*
* Faculty of Technical Sciences/University of Novi Sad, Novi Sad, Serbia
sladicg@uns.ac.rs mbranko@uns.ac.rs ftn_zora@uns.ac.rs

Abstract—Access control is one of key aspects of computer security. Current access control models do not take into account the context of the system and its environment. In this paper, we present a particular context model for the access control system primarily intended for business processes access control. By using a context-sensitive access control, it is possible to define more sophisticated access control policies that cannot be implemented by existing access control models. The proposed context is modeled using Web Ontology Language (OWL) in order to provide: formal representation of context, rich representation of different contextual information, semantic interoperability between different context-aware systems and high degree of inference making.

I. INTRODUCTION

Access control is only one aspect of a comprehensive computer security solution, but it is also one of its most important segments. It provides confidentiality and integrity of information. In the Role-based Access Control (RBAC) model, access to resources of a system is based on a role of a user in the system [1]. The basic RBAC model comprises the following entities: users, roles and permissions, where permissions are composed of operations applied to objects. In RBAC, permissions are associated with roles, and users are made members of roles [1]. This greatly simplifies management of access rights, so the RBAC model has generated great interest in the security community. It is customary to use the role hierarchy [1] to aggregate permissions, i.e. a role is assumed to inherit the permissions assigned to its parent roles in the hierarchy. In addition, the role hierarchy also determines the roles that are available to a user, i.e. a user assigned to a particular role can also activate any subordinate roles in the hierarchy.

The RBAC model has been widely applied to a large number of different information systems in order to meet access control requirements. However, there still exist many problems in the aspect of describing complex workflow access control authorization and constraints. One of the major problems is how to express business character and phase authorization constraints in carrying out the task between the roles and users. Traditional access control models, such as RBAC are passive access control. They do not take into account contextual information, such as processed data, location or time for making access decisions. Consequently, these models are inadequate for specifying the access control needs of many complex real-world workflows. As context data gets involved, the access decision no longer depends on user credentials only, it also depends on the state of the system's environment and the system itself.

In this paper we propose the context model for the context-sensitive access control model for business processes (COBAC) [2, 3]. By using the proposed context model it is possible to define more complex access control policies whose implementation by existing access control models for business processes is not possible or is very complicated. The COBAC's context model can describe rich context information and can be easily extended for specific cases.

II. RELATED WORK

Various definitions of the context have been proposed in the literature [4, 5, 6, 7, 8]. Broadly, the notion of context relates to the characterization of environmental conditions that are relevant for performing appropriate actions in the computing domain. Probably the most widely accepted definition has been provided in [5, 6]:

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

In order to implement the fine-grained access control, many context-sensitive access control models have been proposed. Research in the context-sensitive access control is conducted in two directions, one that analyses the influence of multiple context variables on the access control, and the other that analyses influence of individual context variables on the access control.

Han et al. [9] describe a context-sensitive access control model which consists of a context model, a context-sensitive policy model and a context-sensitive request model. Bao et al. [10] present the Conditional Role Based Access Control (C-RBAC) model. This model relies on RBAC, extends the notion of role by incorporating attributes, and is based on the notion of system context. Covington et al. [11] introduce the notion of an environmental role, and provide a uniform access control framework that can be used to secure context-sensitive applications. In [12] the integration of contextual information with team- and role-based access control is discussed. Filho and Martin [13] suggest the use of context information and its quality indicators to grant access permissions to resources. The use of context-sensitive access control for controlling XML documents is presented in [14, 15, 16]. Strembeck and Neumann [17] deal with context constraints in a RBAC environment. They present an approach that uses special purpose RBAC constraints to base certain access control decisions on context information. In this approach, a context constraint is defined as a dynamic RBAC constraint that checks the actual values of one or more contextual attributes for predefined conditions. If these conditions are met, the corresponding access request can be permitted. Accordingly, a conditional permission is an RBAC...
permission which is constrained by one or more context constraints.

III. THE COBAC MODEL

The COBAC (Context-sensitive access control model for business processes) [2, 3] model explicitly introduces the notion of context, and thus provides a context-sensitive access control for workflow processes. The basic concepts of the COBAC model are presented in Fig. 1. COBAC is based on the standard RBAC model and extends with the following concepts: business process, activity, context and resource category.

The reason for introducing a business process and an activity is to enable efficient use of RBAC, because certain aspects of access control need to be defined for a particular process or its activities. There are cases when a business process can be fully executed within a user's session, but there are also cases where the process execution is distributed through more users' sessions. Therefore, it is necessary that the access control mechanism supports both of these cases. By using business process and activity concepts, it is possible to bind certain access control segments with activities instead of binding them with a current session. We propose two types of business process in the COBAC model: the complex business process and the simple business process. The complex business process is defined as proposed in [18, 19]. It consists of a set of activities between which there is a partial order relation. As there are a lot of “relatively simple” tasks which do not require a workflow for their execution, we introduce the simple business process to represent them (for example, create a notification or create a report). The simple business process consists of only one activity.

Access control policies in the COBAC model can be defined for definitions and instances of business processes. If the policy is defined for a process definition, it will be applied to all instances of that process, while policies defined for an instance will be applied only to this particular instance. By analyzing security requirements for different business processes, we concluded that in the case of complex business processes it is necessary to support both the process level and the process instance level of policies, while in the case of simple business processes it is sufficient to support the process definition level of policies only. Instance-level policies for simple processes can be omitted without the loss of generality.

Since access control may be influenced by some other factors from the system and its environment, the COBAC model is also extended with the notion of context [2, 3]. The influence of the context to access control is accomplished through context conditions in assignment relations (user-role, role-activity, activity-permission) and in constraints. Only if some condition is satisfied, corresponding assignment relation or constraint will be established [2, 3].

During activity execution, it may be necessary to access some resources. In order to execute such an activity, proper permissions which are required for the resources being accessed, are assigned to that activity. The categorization of resources enables the definition of policies for the whole category of resources thus reducing the number of necessary policies [2, 3].

The process of the access control enforcement is performed through four phases [2]:

- role activation,
- creation of activity (task) list that a user can execute,
- verification if the user is allowed to perform the activity in the moment when she/he initiate execution of the give activity, and
- verification if it is allowed to access the required resources during execution of the activity.

![Figure 1. The COBAC model](image-url)
IV. THE CONTEXT MODEL

We chose the Web Ontology Language (OWL) for context modeling due to several reasons. First, it enables a formal representation of context and supports rich representation of different contextual information. Second, it allows the necessary semantic interoperability between different context-aware systems and also enables easy adjustment of the context model for use in different systems. Finally, it provides a high degree of inference making by providing additional vocabulary along with a formal semantics to define classes, properties, relations and axioms.

Basic entities of the context ontology are presented in Fig. 2. This model defines domain ontology for a context in business systems for access control requirements. The domain ontology models only basic concepts and relations between them. When the COBAC model is used in the concrete case, the domain ontology model should be extended with new concepts and relations which are specific for that case.

Two basic classes of the context model (see Fig. 2.) are ContextFact and ContextExpression. The class ContextFact represents a basic context fact, while the class ContextExpression represents the context expression. Different context facts can be classified as ContextFact specializations.

The Actor class is used for representing different actors of events. Different activities are modeled by the Action class, and resources are modeled by the Resource class. Location is described by the Location class. The Time class is used for modeling time factor in the model. Different purposes in this model are represented by Purpose class, while means are described with the Means class.

The context expression (the class ContextExpression) represents a semantic binding of previously listed concepts, and it is based on 7 semantic dimensions (relations). Each aforementioned context fact creates one semantic dimension. Actually, the context expression describes events that took place and the conditions under which these events occurred. We extend five semantic dimensions defined in [20, 21] (“who”, “what”, “where”, “when” and “how”) with the notion “why” which defines the purpose and with the notion “related” which defines the relation between different context expressions. We also define two specializations of the “what” concept. The specialization “what action” is used for defining the “what” relation with an action, while the specialization “what resource” is used for defining the “what” relation with a resource.

The context expression definition is presented in Table I. In our definition, the context expression must contain at least one “who” and one “what” relation. We add this restriction because it is necessary that the context expression contains the information who/what did something, and what he/it did in order to describe an event.
V. THE CONTEXT CONDITION

We define the context condition as a logical expression which may consist of queries for searching context ontology database (like SPARQL), context functions, logical operators (¬, ∧, ∨) and comparison operators (<, ≤, ≥, ≠). The context functions are used to retrieve some current information from a system, like who is the current user, or what activity is being currently executed. The context condition defined in the EBNF notation is presented in Table II.

VI. SOFTWARE ARCHITECTURE

The main classes used for the context representation are presented in Fig 3. The context module of the COBAC system consists of two segments. One of segments provides support for context sensitive access control, while the other segment enables context acquisition.

The rest of the COBAC system accesses to the context subsystem using a proper implementation of the ContextService interface. The class OWLContextService is
used when context is modeled using OWL. The context entities are modeled with implementations of the ContextEntity interface. The context fact is represented by the ContextFact class; the context relation is modeled with ContextRelation, and the class ContextExpression is used to describe the context expression.

Acquisition of context data is performed using context sensors which are modeled with the ContextSensor class. Acquisition can be performed by monitoring the state of a system (StateContextSensor) or using certain events i.e. using Event Condition Action (ECA) rules (ECAContextSensor). When events are used for the context acquisition, the class ContextEvent represents an event, while the class ContextEventListener models a proper listener for the event. The data acquisition form a software component is implemented with the classes SoftwareECAContextSensor and SoftwareStateSensor.

The context data collected by the context sensors can be stored in the context repository using ContextStorageService or it can be passed to context aggregators (the class ContextAggregator) which create derived context data based on the collected context data. The created derived data is also stored in the context repository. The class AggregatorRegistry provides information to which aggregators, context sensors should send their data.

The context condition, used for defining context dependant access control policies, is represented by the ContextCondition class presented in Fig 4. The query for searching context data is modeled with the ContextQuery class. SPARQLContextQuery enables creation of search criteria using the SPARQL language. The context functions are represented by the ContextFunction implementation. The logical operators in the context condition are modeled with the LogicalOperator class, while the comparison operators are modeled with the ComparisonOperator class.

![Figure 3. The class diagram of the context subsystem](image)

![Figure 4. The context condition class diagram](image)
A possible solution for this problem is to use context-sensitive access control. The context is used to handle access control requirements which depend on factors from the system and from the system’s environment. In this paper we present the context model for the Context-sensitive access control model for business processes (COBAC).

The presented context model can describe rich context information and can be easily extended for specific cases. It also allows the semantic interoperability between different context-aware systems. The context information can be used in all segments of the COBAC model. Because of this, the COBAC model can be easily adopted for use in different business processes. The proposed context model is modeled using Web Ontology Language (OWL) in order to provide: formal representation of context, rich representation of different contextual information, semantic interoperability between different context-aware systems and high degree of inference making.

The context model’s prototype implementation is verified on the access control enforcement for a real workflow system based on documents - national petty offense trial proceedings [2, 3], by which the practical value of the proposed model is proved.

Our experience in using the COBAC prototype in the case study presented in [2, 3] showed that defining contexts using ontologies can be difficult for the most information security officers due to lack of experience in the use of ontologies. Therefore research on a DSL (domain specific language) for describing context in order to enable easy context definition is in progress.

The proposed context model does not consider the quality of the context information, like accuracy, reliability, etc. This information can be very important for access control in certain cases. A further direction in development of the presented context model includes the use of the context quality in modeling context.

REFERENCES