Journal of Computational Science 5 (2014) 653–663

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Assuring consistency in mixed models

Ramzi A. Haraty*, Mirna F. Naous, Azzam Mourad

Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon

ARTICLE INFO

Article history:
Received 10 September 2013
Received in revised form 8 January 2014
Accepted 21 February 2014
Available online 3 March 2014

Keywords:
Role based access control
Clark Wilson
Consistency
Alloy

ABSTRACT

Information systems security defines three properties of information: confidentiality, integrity, and availability. These characteristics remain major concerns throughout the commercial and military industry. Ordinary users have taken these features as basis for their businesses. Furthermore, users may find it necessary to combine policies in order to protect their information in a suitable way. However, inconsistencies may arise as a result of implementing multiple secrecy and privacy models; and therefore, render these services unsecure. In this paper, we propose an approach to detect and report inconsistencies when choosing mixed models for integrity and security. It is based on specifying the policies in first order logic and applying formal analysis. We demonstrate the feasibility of our proposition by applying it to the Clark Wilson and role based access control models. We use the Alloy language and analyzer to formalize the mixed model and check for any inconsistencies.

1. Introduction

The goal of information systems is to control or manage the access of subjects (users, processes) to objects (data, programs) [11]. This control is governed by a set of rules and objectives called a security policy. Data integrity is defined as "the quality, correctness, authenticity, and accuracy of information stored within an information system" [4]. Systems integrity is the successful and correct operation of information resources. Integrity models are used to describe what needs to be done to enforce the information integrity policies. There are three goals of integrity:

- Prevent unauthorized modifications,
- Maintain internal and external consistency, and
- Prevent authorized but improper modifications.

Combining multiple secrecy and privacy policies in the same service may cause inconsistencies. An inconsistency or contradiction may undermine the proper functioning of the service creating instances where some users may be allowed access to information according to a rule, while not allowed according to another. Clearly, it is important that inconsistencies be detected before the online deployment of the system. Furthermore, users may not understand the implications of adding or removing policies, which may grant rights that threaten privacy, or revoke rights that are instrumental to information sharing.

Before developing a system, one needs to describe formally its components and the relationships between them by building a model. The model needs to be analyzed and checked to figure out possible bugs and problems. Thus, formalizing integrity security models helps designers to build a consistent system that meets its requirements and respects the three goals of integrity. This objective can be achieved through the Alloy language and its analyzer.

Alloy is a structural modeling language for software design. It is based on first order logic that makes use of variables, quantifiers and predicates (Boolean functions) [6]. Alloy, developed at MIT, is mainly used to analyze object models. It translates constraints to Boolean formulas (predicates) and then validates them using the Alloy Analyzer by checking code for conformance to a specification [18]. Alloy is used in modeling policies, security models and applications, including name servers, network configuration protocols, access control, telephony, scheduling, document structuring, and cryptography. Alloy's approach demonstrates that it is possible to establish a framework for formally representing a program implementation and for formalizing the security rules defined by a security policy, enabling the verification of that program representation for adherence to the security policy.

There are several policies applied by systems for achieving and maintaining information integrity. In this paper, we focus on the role based access control and Clark Wilson and to show how they can be checked for consistency or inconsistency using the Alloy language and the Alloy Analyzer.

* Corresponding author. Tel.: +961 1867620; fax: +961 1867098.
E-mail address: rharaty@lau.edu.lb (R.A. Haraty).

http://dx.doi.org/10.1016/j.jocs.2014.02.009
1877-7503/

The remainder of this paper is organized as follows: Section 2 provides the literature review. Section 3 discusses the role based access control security model. Section 4 discusses the Clark Wilson security model. Section 5 present the mixed model and Section 6 concludes the paper.

2. Literature review

Modeling and validation has been used extensively in the literature [3–13]. Hassan and Logrippo [20] proposed a method to detect inconsistencies of multiple security policies mixed together in one system and to report the inconsistencies at the time when the secrecy system is designed. The method starts by formalizing the models and their security policies. The mixed model is checked for inconsistencies before real implementation. Inconsistency in a mixed model is due to the fact that the used models are incompatible and cannot be mixed.

Zao et al. [9] developed the RBAC schema debugger. The debugger uses a constraint analyzer built into the lightweight modeling system to search for inconsistencies between the mappings among users, roles, objects, permissions and the constraints in a RBAC schema. The debugger was demonstrated in specifying roles and permissions according and verifying consistencies between user roles and role permissions and verifying the algebraic properties of the RBAC schema.

Hassan et al. [21] presented a mechanism to validate access control policy. The authors were mainly interested in higher level languages where access control rules can be specified in terms that are directly related to the roles and purposes of users. They discussed a paradigm more general than RBAC in the sense that the RBAC can be expressed in it.

Shaffer [1] described a security Domain Model (DM) for conducting static analysis of programs to identify illicit information flows, such as control dependency flaws and covert channel vulnerabilities. The model includes a formal definition for trusted subjects, which are granted privileges to perform system operations that require mandatory access control policy mechanisms imposed on normal subjects, but are trusted not to degrade system security. The DM defines the concepts of program state, information flow and security policy rules, and specifies the behavior of a target program.

Misic and Misic [8] addressed the networking and security architecture of healthcare information system. This system includes patient sensor networks, wireless local area networks belonging to organizational units at different levels of hierarchy, and the central medical database that holds the results of patient examinations and other relevant medical records. In order to protect the integrity and privacy of medical data, they proposed feasible enforcement mechanisms over the wireless hop. Haraty and Naous [15–17] also modeled the clinical information systems policy. The authors used the Alloy formal language to define these models and the Alloy Analyzer to validate their consistency.

Haraty et al. [16] showed how secrecy policies can be checked and inconsistency by modeling the Chinese Wall Model [5], Biba Integrity Model [10], Lipner Model [19] and the Class Security Model [2]. In their work, they listed the ordered security classes (Top Secret, Secret, Confidential, and Unclassified) and their compartments (nuclear, technical, and biological) and defined those using signatures. Then, the possible combinations and the relationships between classes and compartments were specified. Facts were used to set the model constraints and to prove that the model is consistent. In the Biba Integrity model, the authors listed the subject security clearance and the object security classes and then modeled the constraints of how subjects can read/write objects based on "NoReadDown" and "NoWriteUp" properties. In [14], the author presented a comparison of different secure systems with their access control policies.

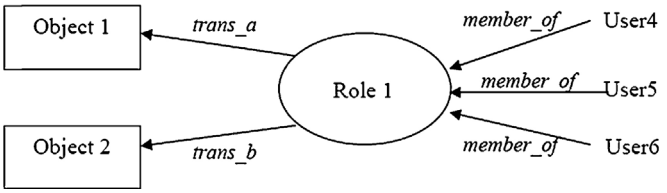

Fig. 1. Roles relatioship.

3. Role-based access control

In many organizations, "the end users do not own the information for which they are allowed access" [7]. However, the company is the actual owner of system objects as well as the programs that process it. Control is often based on employee functions and access control decisions are often determined by the users' roles. This suggests associating access with particular role of the user.

Since access to system modules is not tied to particular individual, role-based access control (RBAC) is used by many organizations to protect data integrity and restrict access to information based on users' roles. RBAC model defines the following entities:

- Role *r* is a collection of job functions. It is a set of transactions that a user or set of users can perform within an organization.
- Transactions are allocated to roles by a system administrator. Each role is authorized to perform one or more transactions *t*.
- Subject s has roles in the system. The active role of a subject is the role that is currently performing.

Fig. 1 shows Role 1 containing users 4, 5 and 6 as members. Users of Role 1 can execute transactions trans_a and trans_b on object 1 and object 2.

Ferraiolo in [7] determined the formal description of RBAC in terms of sets and relations as follows:

- For each subject, the active role is the one that the subject is currently using: *AR*(*s*: *subject*) = {*the active role for subject s*}.
- Each subject may be authorized to perform one or more roles:
- RA(s: subject) = {authorized roles for subject s}.
- Each role may be authorized to perform one or more transactions: *TA(r: role)* = {*transactions authorized for role r*}.
- Subjects may execute transactions. The predicate *exec(s,t)* is true if subject *s* can execute transaction *t* at the current time, otherwise it is false: *exec(s: subject, t: tran)* = *true iff subject s can execute transaction t.*

Accordingly, three basic rules are required in RBAC model:

- *Role assignment*: A subject can execute a transaction only if the subject has selected or been assigned a role. However, all active users are required to have some active role.
- *Role authorization*: A subject's active role must be authorized for the subject with role assignment; this rule ensures that users can take on only roles for which they are authorized.
- *Transaction authorization*: A subject can execute a transaction only if the transaction is authorized for the subject's active role.

Therefore, and according to RBAC model, a subject *s* can execute a transaction *t* if it has an active role *r* and its role is authorized to execute the transaction *t*. Also, RBAC can model the separation of duty rule since users in some roles cannot enter other roles. This means that users cannot perform the job functions of other roles.