Role-Based Access Control Modeling and Validation

Ramzi A. Haraty and Mirna Naous
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: rharaty@lau.edu.lb

Abstract---**Information systems security defines three properties of information: confidentiality, integrity, and availability. These characteristics remain major concerns throughout the commercial and military industry. In this work, we focus on the security aspect of commercial security applications by exploring the nature and scope of the famous security policy - the Role Based Access Control Policy. We model it and check its consistency using the Alloy Analyzer.**

Keywords: **Role Based Access Control Security Model, Consistency, and Integrity.**

I. INTRODUCTION

The goal of information systems is to control or manage the access of subjects (users, processes) to objects (data, programs). 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" [1]. 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.

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) [2]. 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 [3]. 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 [4] and to show how it can be checked for consistency or inconsistency using the Alloy language and the Alloy Analyze.

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, and section 4 concludes the paper.

II. LITERATURE REVIEW

Hassan and Logrippo [5] 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. [10] 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 [11] 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 [12] 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 Boss [13] showed how secrecy policies can be checked for consistency and inconsistency by modeling the Chinese Wall Model [14], Biba Integrity Model [15], Lipner Model [16] and the Class Security Model [17]. Haraty and Naous [18] 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. 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 [19], the authors presented a comparison of different secure systems with their access control policies.

III. RBAC IMPLEMENTATION

In many organizations, "the end users do not own the information for which they are allowed access" [12]. 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.

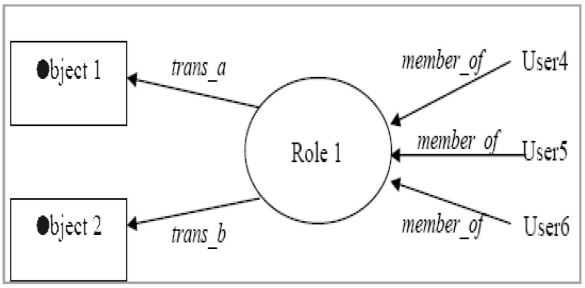

Figure 1. Roles Relatioship

Ferraiolo in [12] determined the formal description of RBAC in terms of sets and relations as follow:

- 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. Moreover, some roles subsume others. This defines a hierarchy of roles. Granting access to a role *r* implies that access is granted for all roles containing *r*.

A. Role-Based Access Control Implementation

In order to implement the RBAC model, a procurement management system (PMS) is used to demonstrate model consistency. Table 1. summarizes available roles in the system and their main job functions regardless the name of users playing these roles.

TABLE 1. PROCUREMENT AMANGEMENT SYSTEM ROLES

Role		Job Description
Employee	**RE**	Fills new purchase request and submits it to procurement officer for processing.
Procurement Officer	**RP**	Checks submitted requests, contacts suppliers then fills purchase orders.
Supervisor	**RS**	Reviews purchase orders then submit them to manager for approval. Reviews the delivery of purchased items then submits it to manager for approval.
Manager	**RM**	Approves purchase orders Approves deliveries. Approves Payments.
Store Keeper	**RK**	Issues deliveries of purchased items.
Accountant	**RA**	Inserts payments.

Table 2. lists the transactions that can be performed in the procurement management system:

TABLE 2. PROCUREMENT MANAGEMENT SYSTEM TRANSACTIONS

Transaction	Description
TIR	Insert purchase request.
TIPo	Insert purchase order.
TRPo	Review purchase order.
TAPo	Approve purchase order.
TID	Issue delivery
TRD	Review delivery
TAD	Approve delivery
TIP	Insert payment
TAP	Approve payment

There are six roles in the system as described in Table 1. The manager role is the top role in the system. This role subsumes all other system roles. Thus, the manager is allowed to perform the tasks assigned to supervisor, accountant, procurement officer, store keeper, and employee roles. Furthermore, the supervisor role can do the work of procurement user, store keeper and the employee roles. However, it cannot handle the accountant tasks since accountant role is directly under the manager role in the hierarchy. Accountant, Procurement officer and store keeper are 3 different roles in the system and they have the ability to perform their jobs functions as well as the employee functions.

Table 3. specifies system users, their roles and the transactions that are authorized to execute based on the defined roles. For example, in the PMS, Mirna is given the employee role only. The employee role can execute TIR (insert purchase request transaction). Accordingly, Mirna can execute TIR on behalf of her role. Any other active user given the RE role can execute TIR since the execution is assoicated with the particular job not the user. However, the supervisor role represented by Fadi Feghali is authorized to execute maily TRPo and TRP. But since Fadi's role is above the store keeper, procurement officer and employee roles, Fadi or any active user given this role can execute the transactions authorized by RP, RK and RE (i.e., TIP, TIPo, TID).

TABLE 3. PROCUREMENT MANAGEMENT SYSTEM USER ROLES

User name		Role	Transaction
Mirna Naous	UM	RE	TIR
Hossam Abu Laban	UH	RP	TIR -TIPo
Fadi Feghali	UF	RS	TIR - TIPo - TRPo - TID - TRD
Nagy Karkour	UN	RM	TIR - TIPo - TRPo - TAPo - T ID - TRD -TIP- TAP
Rehab Salloum	UR	RA	TIR - TIP
Jaafar Hashem	UJ	RK	TIR -TID

The procurement management system can be represented using the Alloy language. The following sections provide detailed description of Alloy code.

- Section 1 declares the set of roles and transactions. According to the system there are six roles defined in table 1. The supervisor role RS is under the manager role RM, the accountant role RA is under the RM, the procurement officer role RP is under RS, the store keeper role RK is under RS and RE is under RA, RK and RP.

```
//declaration of roles
abstract sig Roles{}
abstract sig Trs{executedby:some Roles}

one sig RM extends Roles{}//Role:Manager
one sig RS extends Roles{under:RM} //Role: Supervisor
one sig RA extends Roles{under:RM}//Role: Accountant
one sig RP extends Roles{under:RS}//Role:Procurement Officer
one sig RK extends Roles{under:RS}//Role:Store keeper
one sig RE extends Roles{under:RA+RK+RP} //Role: Employee
```

Section 1 – Procurement Management System Declaration.

- Section 2 defines the procurement system transactions as part of Trs set.

```
//declaration of transactions
one sig TIR,TIPo,TIP,TID,TRPo,TRD,TAPo,TAD,TAP extends Trs {}
```

Section 2 – Procurement Management System Transactions

- Section 3 specifies system users and their roles. For instance user Mirna UM is playing the employee role in the system, Hossam UH is playing the role of procurement officer RP, Fadi UF is playing the supervisor role RS, and so on.

```
//declaration of users
one sig UM in RE{}//user:Mirna
one sig UH in RP{}//user:Hossam
one sig UF in RS{}//User Fadi
one sig UN in RM{}//User Nagy
one sig UJ in RK{}//user Jaafar
one sig UR in RA{}// user rehab
```

Section 3 – Procurement Management System – System Users.

- Section 4 restricts the execution of transactions to certain roles. Table 3 determines the roles and the transactions that are allowed to execute in the procurement system. The transaction approve purchase order TAPo, approver payment TAP and approve delivery TAD need to be executed by the manager role only, other roles are not allowed to perform such action.

```
fact {

// Transaction approve PO, approve delivery, approve payment can be accessed by RM only
TAPo.executedby!=RE
TAPo.executedby!=RK
TAPo.executedby!=RP
TAPo.executedby!=RA
TAPo.executedby!=RS

TAD.executedby!=RE
TAD.executedby!=RK
TAD.executedby!=RP
TAD.executedby!=RA
TAD.executedby!=RS

TAP.executedby!=RE
TAP.executedby!=RK
TAP.executedby!=RP
TAP.executedby!=RA
TAP.executedby!=RS
```

Section 4 - Procurement Management System Transactions Constraints (Part 1)

- Section 5 states that the review purchase order transaction TRPo and the review delivery TRD cannot be accessed by employee RE, store keeper RK, procurement officer RP and accountant RA. Thus, TRPo and TRD can be executed by RS and since RS is under RM in the hierarchy then RM can execute them also.

```
//transaction Review PO and review delivery can be accessed by RS only or RM since RS under RM
TRPo.executedby!=RE
TRPo.executedby!=RK
TRPo.executedby!=RP
TRPo.executedby!=RA

TRD.executedby!=RE
TRD.executedby!=RK
TRD.executedby!=RP
TRD.executedby!=RA
```

Section 5 - Procurement Management System Transaction Constraints (Part 2)

B. Role-Based Access Control and Alloy Analysis

As shown previously, the Alloy analyzer helps checking system validity by generating system Meta model and by generating instances of the system based on its facts and predicates. Fig. 2 displays the procurement management system meta model generated by the Alloy system.

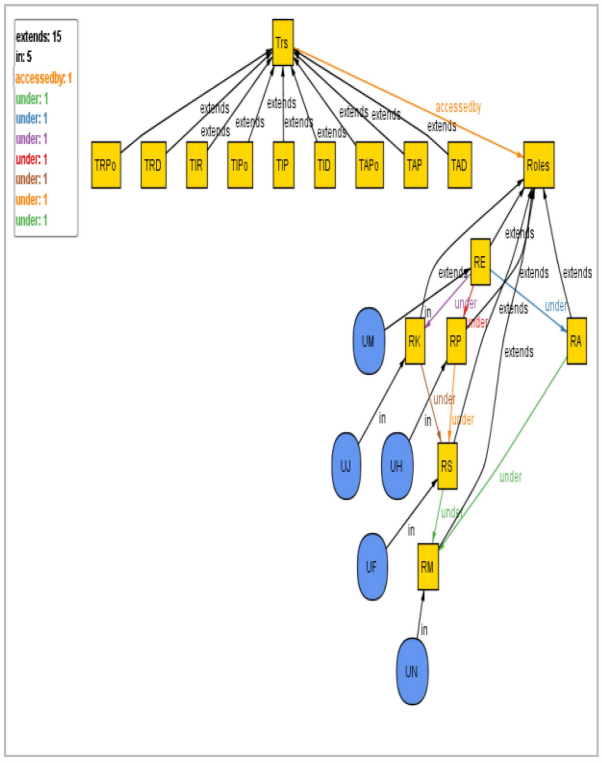

Figure 2. Procurement management system meta model.

The model maps the codes written in the previous sections into a graphical model. It shows the different roles of the system. Additionally, the figure displays the system users attached to their roles, as well as the system transactions. However, the meta model does not show any constraints. Executing the system using the Alloy analyzer will generate instances based on defined constraints.

Testing system consistency is done by running the system predicates and generating possible instances then validating them. In order to test the constraints specified in the fact procedure, a predicate is written is written and executed.

- Section 6 declares an empty predicate used to test the system consistency based on the defined facts. Executing the example yields the output shown in fig 3., which shows that “an instance is found” and “Predicate is consistent.

```
// run example to show consistencies/ inconsistencies
pred example(){
}
run example
```

Section 6 - Procurement Management System Predicate

```
Executing "Run example"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  140 vars. 66 primary vars. 137 clauses. 110ms.
  Instance found. Predicate is consistent. 109ms.
```

Figure 3. Procurement Management System Consistent Alloy Analyzer Output

Clicking the link “Instance” will yield fig. 4. The generated instance demonstrates the consistency of the system.

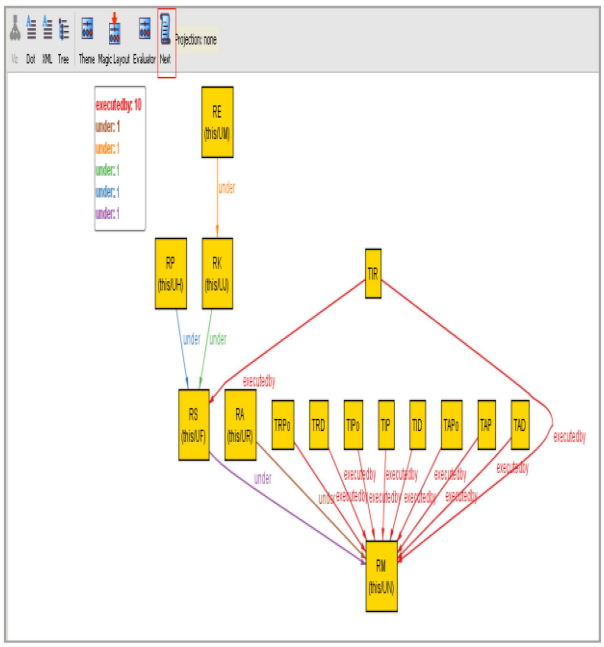

Figure 4. Procurement Management System Instance.

However, specifying a wrong predicate, such as stating that employee role RE can execute the insert Po transaction will cause inconsistency.

IV. CONCLUSION

In this paper, we presented the Role-Based Access Control. We used system examples based on the defined security model. We formalized the system according to the model then checked its consistency and inconsistency. Since Alloy allows expressing systems as set of logical constraints in a logical language based on standard first-order logic, we used it to define the system and its policy. However, when creating the model we specified the system users and subjects then Alloy compiles a Boolean matrix for the constraints, and we asked it to check if a model is valid, or if there are counterexamples.

REFERENCES

[1] C. Summers, "Computer security: threats and safeguards". McGraw Hill. New York, 1997.

[2] D. Jackson, "Alloy 3.0 reference manual". Retrieved on January 24, 2014 from: http://alloy.mit.edu/reference-manual.pdf.

[3] R. Seater and G. Dennis, "Tutorial for Alloy Analyzer 4.0". Retrieved on January 24, 2014 from: http://alloy.mit.edu/tutorial4.

[4] R. Anderson, "A security policy model for clinical information systems". Proc. of the IEEE Symposium and Security and Privacy, USA, 1996.

[5] W. Hassan and L. Logrippo, "Detecting inconsistencies of mixed secrecy models and business policies". University of Ottawa, Canada, Technical Report, 2009.

[6] L. Bell and E. LaPadula, "Secure computer systems: mathematical foundations". Technical Report 2547, Volume I, The MITRE Corporation, 1976.

[7] D. F. Ferraiolo and D. R. Kuhn, "Role-Based Access Control". Proceedings of the 15th National Computer Security Conference, Baltimore, 1992.

[8] J. Viega and D. Evans, "Separation of Concerns for security". Proceedings of the Workshop on Multi-Dimensional Separation of Concerns in Software Engineering. Limerick, Ireland, pp. 126-129, 2000.

[9] J. Zao, W. Hoetech, J. Chu and D. Jackson, "RBAC schema verification using lightweight formal model and constraint analysis". Proceedings of 8th ACM Symposium on Access Control Models and Technologies, Boston, MA, USA, 2003.

[10] W. Hassan, L. Logrippo and M. Mankai, "Validating access control policies with Alloy". Proceedings of the Workshop on Practice and Theory of Access Control Technologies, Quebec, Canada, pp. 17-22, 2005.

[11] A. Shaffer, M. Auguston, C. Irvine and T. Levin, "A security domain model to assess software for exploitable covert channels". Proceedings of the ACM SIGPLAN Third Workshop on Programming Languages and Analysis for Security, Tucson, Arizona. USA, pp. 45-56, 2008.

[12] J. Misic and V. Misic, "Implementation of security policy for Clinical Information Systems over wireless sensor networks". Ad Hoc Networks Journal 5, pp. 134-144, 2006.

[13] R. A. Haraty, N. Boss and M. Naous, "Modeling and validating confidentiality, integrity, and object oriented policies using Alloy". Security and Privacy Preserving in Social Networks. Springer. ISBN 978-3-7091-0893-2, 2013.

[14] D. Brewer and M. Nash, "The Chinese Wall Security policy". Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, USA, pp. 206-214, 1989.

[15] K. J. Biba, "Integrity Considerations for Secure Computer Systems". Technical Report MTR-3153, The MITRE Corporation, 1977.

[16] S. B. Lipner, "Non-discretionary controls for commercial applications". Proceedings of the IEEE Symposium on Security & Privacy, Oakland, 1982.

[17] B. Panda, W. Perrizo and R. A. Haraty. "Secure transaction management and query processing in multilevel secure database systems". Proceedings of the ACM Symposium on Applied Computing. Phoenix, AZ. April 1994.

[18] R. A. Haraty and M. Naous, "Modeling and validating the Clinical Information Systems Policy using Alloy", Proceedings of the Second International Conference on Health Information Science. Lecture Notes in Computer Science - Springer. London, England. March 2013.

[19] R. A. Haraty, "C2 Secure Database Management Systems – A Comparative Study". Proceedings of the Symposium on Applied Computing. San Antonio, TX, USA, pp. 216–220, 1999.