Research Journal of Applied Sciences, Engineering and Technology 7(18): 3786-3790, 2014
DOI:10.19026/rjaset.7.734
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 26, 2013 Accepted: December 21, 2013 Published: May 10, 2014

Research Article

Secure State UML: Modeling and Testing Security Concerns of Software Systems Using UML State Machines

S. Batool and S. Asghar
Institute of Information Technology, University of Arid and Agriculture Rawalpindi, Pakistan

Abstract: In this research we present a technique by using which, extended UML models can be converted to standard UML models so that existing MBT techniques can be applied directly on these models. Existing Model Based Testing (MBT) Techniques cannot be directly applied to extended UML models due to the difference of modeling notation and new model elements. Verification of these models is also very important. Realizing and testing non functional requirements such as efficiency, portability and security, at model level strengthens the ability of model to turn down risk, cost and probability of system failure in cost effective way. Access control is most widely used technique for implementing security in software systems. Existing approaches for security modeling focus on representation of access control policies such as authentication, role based access control by introducing security oriented model elements through extension in Unified Modelling Language (UML). But doing so hinders the potential and application of MBT techniques to verify these models and test access control policies. In this research we introduce a technique secure State UML to formally design security models with secure UML and then transform it to UML state machine diagrams so that it can be tested, verified by existing MBT techniques. By applying proposed technique on case studies, we found the results that MBT techniques can be applied on resulting state machine diagrams and generated test paths have potential to identify the risks associated with security constraints violation.

Keywords: Model based testing, object constraint language, role based access control, unified modeling language

INTRODUCTION

In the software industry, the use of models for testing the software systems before its implementation is called Model Based Testing (MBT), (Lindholm *et al*., 2006). Model based testing is basically applied for testing functional requirements of the system, but non functional requirements such as performance, usability and security are also important for any software system. Many efforts are now being put forward by researchers such as Lodderstedt *et al*. (2002), Jurjen (2002), Gray (2004) and Mariscal *et al*. (2005), to address different non functional aspects such as security using MBT.

Jurjen (2002) introduced UMLsec as a UML profile for modeling security critical systems. UMLsec uses stereo types, tags and constraints to model security such as availability and integrity. Dynamic view of RBAC is presented Mariscal *et al*. (2005) presents new diagrams are introduced for specifying security policies by combination of the basic security models i.e., DAC, MAC and RBAC. It is also not easy to accommodate the changes in the overall design of the system that will happen due to integration of security model in the overall functional model of the system. Introduces an approach that uses UML sequence and state machine diagrams in combination to model the system functionalities and security threats.

The proper identification of threats is a difficult task and the threats that are not relevant to the control flow may be missed due to the fact that state machine diagram is used to model the control based transition of system states. Ceneys *et al*. (2009) analysis the two well known UML based modeling techniques i.e.

Secure UML and UMLsec in context of modeling RBAC. Raimundas and Dumas (2011) presented the comparison of Secure UML and UMLsec in the context of modeling RBAC. The authors presents later the transformation rules for converting Secure UML models to UMLsec and vice versa. The idea behind this was to utilize the modeling capabilities of both the languages for RBAC based system.

The transformation rules are based on a simple modeling example. The need is highlighted by the author for application of the transformation rules to a complex system to check its validity. Moreover to move from one extension of UML to another extension is not a healthy approach due to its incompatibility with standard UML models and MBT testing techniques and tools and less expertise are available in the extended

Corresponding Author: S. Batool, Institute of Information Technology, University of Arid and Agriculture Rawalpindi, Pakistan

modeling approaches when compared with the expertise that are available in UML.

In this research we aim to introduce a technique for transforming the secure UML models to UML state machine diagrams so that we can apply the existing MBT techniques on extended UML models along with security requirements, for verification of the required security requirements properly modeled or not to avoid future security risk.

MATERIALS AND METHODS

In this section we present the proposed approach for modeling and testing security. Through study of existing literature we highlighted the need of testing and verification of the extended UML models used for modeling security policies. We introduce a technique to enable the application of existing MBT techniques on these models. We use secure UML (Lodderstedt *et al*., 2002), an extension of UML for modeling RBAC policies. Secure UML uses stereotypes of class diagram for modeling of the RBAC concepts. We transformed secure UML models to UML state machines so that it can present the dynamic view of the secure UML models.

In the following Fig. 1 we present the overall process of the proposed technique secure State UML. Ellipses are used to show the activities that are performed and boxes are used to present the input to these activities, Boxes are used to show the output produced in result of these activities.

Components of secure state UML: We present the details of each activity involved in the secure state UML modeling and testing process.

System modeling with secure UML: The modeling capabilities of secure UML (Lodderstedt *et al*., 2002). Secure UML provides the elements for modeling RBAC policies.

The authorizations constraints of secure UM are used to model the pre conditions on access to system resources are very strong feature to control access to system resources. The following steps are carried out to model the system in secure UML:

- Identify users of the system and model each user as a class of secure UML user.
- Identify the roles and model them as classes secure UML.role and associate users with their specific roles by assignment links.
- Identify and associate the permissions of the roles and model them in secure UML.permission class.
- Model the resources in secureuml.resource class and define the associated methods.
- Authorization Constraints (AC) are used to apply the access control on resources based on the security policies. AC can be applied on resources directly or through permissions to roles.

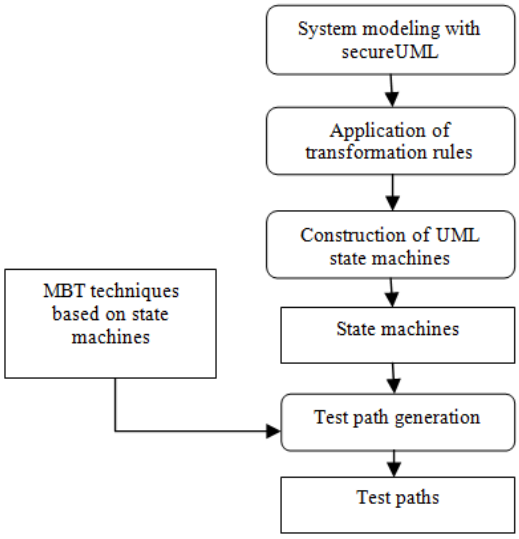

Fig. 1: Process flow of secure state UML

Mapping of secure UML models to UML state machines: In transformation rules, we specify mapping of secure UML modeling elements with different elements of UML state machines. A behavioral state machine can be used to model the behavior of instances of classes (UML superstructure specification 2.0). Secure UML model is presented in form of class diagrams stereotypes. Based on the references of UML superstructure 2.0 we can create state machine diagrams from Secure UML models along with the security information that is specified in these models. For instance, we map Authorization Constraints (AC) of secure UML to guard conditions in state machine diagram. Different elements of secure UML can be mapped to state machine model as we state in the Table 1. The first two columns presents the mapping of RBAC concepts to secure UML (Matulevicius and Dumas, 2011). We further extend and provide the mapping of secure UML modeling elements to be used for constructing UML state machine diagrams of the proposed technique secure State UML. Most of the techniques for testing and modelling Object Oriented (OO) systems are focused on use of state machines and class diagram (Thapa *et al*., 2010). Secure UML is based on class diagram stereotypes. Transformation to State machine gives its dynamic view and more testing expertise and tool support.

Construction of UML state machine diagrams: Through transformation rules we map each element in secure UML model to state machine diagram. In this way, we are able to construct a complete state machine diagram which can become an alternate depiction

Table 1: Mapping secure UML model elements to secure state UML

RBAC concepts	Secure UML model construct	Secure state UML
User	Class stereotype	User (operation attribute at verification state)
User, role assignment	<<secuml.user>> dependency stereotype	Verification state
Role	<<assignment>> class stereotype	Role (operation attribute at verification state)
Permission assignment	<<Secuml.role>> association class stereotype	Requested event trigger when the associated guard if any is evaluated
Operations	<<secuml.permission>> operations of class	Requested states, entry/do/exit actions
Object	<<secuml.resource>> class stereotype	Context of class (from secure UML. resource) where state machine execution occurs
Permissions	Authorization constraint in OCL/preconditions	Guard condition/condition predicates/pre conditions

of secure UML model to present the dynamic view of secure UML. Steps to be followed for state machine diagram construction are given bellow:

- Start state machine diagram from initial pseudo state, Initial state will be created as verification state that will verify the user and role assignment of secure UML.
- Request the resource class operation that will be placed as event in the state transition.
- Put the Requested operation name as event and place the precondition if any in guard section of the transition. In case guard is true, move to the target state, else the request rejected state will be constructed and in both cases next transition will lead to the final state of the state machine diagram.

Application of existing MBT technique: Model based testing is an evolving field in the industry for testing. Model based testing techniques are introduced to cover different testing levels such as system level testing, integration level, component level and regression level testing (William, 2006). We transform secure UML models i.e., class diagram stereotypes to UML state machines. By transformation of secure UML models to UML state machines we are able to apply existing MBT techniques on the resulting models for verification of the security requirements. In the Secure State UML for generation of test paths from resulting state machines diagrams. We use test path generation technique presented by Shaukat *et al*. (2006). The output of this activity is the abstract Test Paths.

Test path generation: Application of Existing MBT techniques on State Machines results in generation of test paths. These test paths can be further transformed to executable test cases and run on the system under test to find the unsecure states of the system. These test paths have potential to save cost and time and identify the security design errors in the system as well. Each test path contains set of actions along with pre condition and source state. Execution of action leads to destination state on the approval of associated guard condition. The syntax of test paths and representation of its elements are described below by (Shaukat *et al*., 2006):

[Condition] message $ class_name @ current_state → [guard] destination_state

The elements included in the test path format are explained in the following lines.

Condition: Condition that should be met before execution of message/operation

Message: Operation that transform state machine form one state to another

Class name: Represent class of states

Current_state: State where execution of message/ operation takes place

Destination state: State that appears on execution of message/operation.

RESULTS AND DISCUSSION

In this section we present validation and verification of the proposed technique by applying it on the case study. Comparison of the results is also presented with the previous techniques (Matulevicius and Dumas, 2011). Ttransformation technique for converting secure UML models to UMLsec and vice versa. Description of the case study is presented below.

Case study:

Meeting scheduler system: We illustrate meeting scheduler system presented by Matulevicius and Dumas (2011). The requirements of the system are stated below:

- There are two types of roles in the system, meeting initiator and meeting participants.
- Meeting initiator is authorized to insert and update meeting schedule.
- Meeting participants are permitted to view the details of the meeting in which they are participating.

The secure UML model of the system is presented in the following Fig. 2. Meeting initiator is authorized to enter agreement details and change it as well, meeting participant has the permission to view the

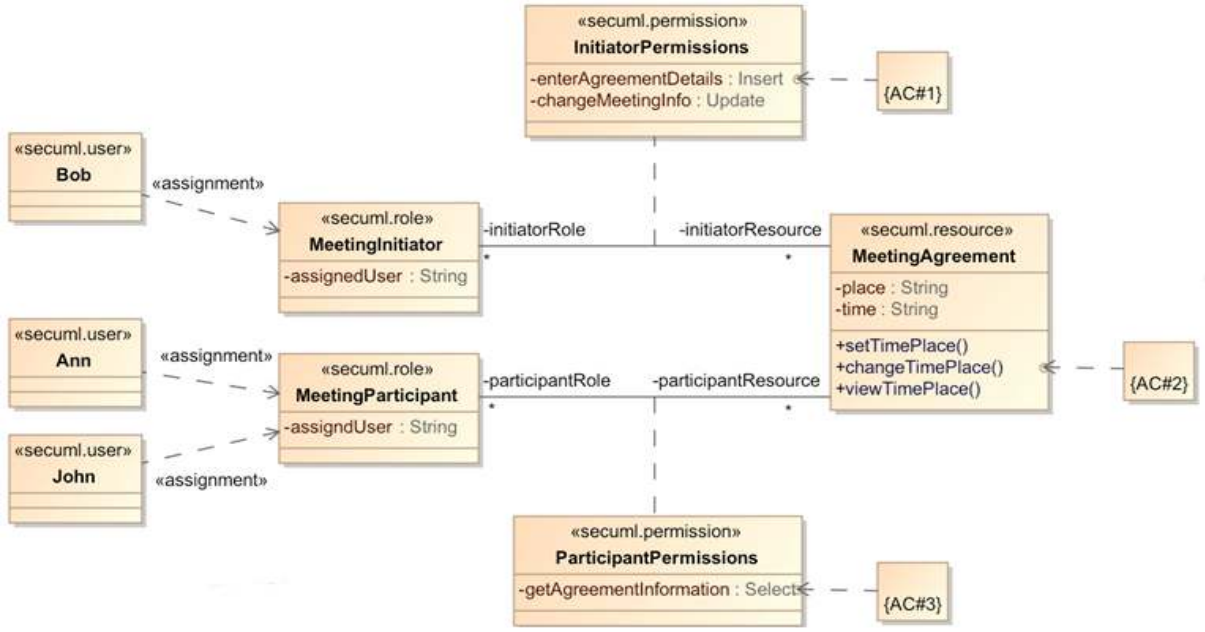

Fig. 2: Secure UML model of the meeting scheduler system

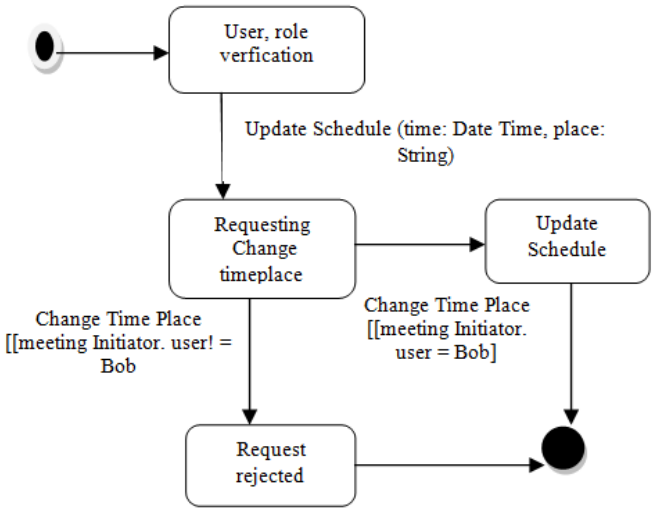

Fig. 3: Transformed state machine diagram of change time place ()

Table 2: Mapping secure UML model elements to UML state machines

Secure UML model element	Mapping element of state machines
Bob	User (operation attribute at verification state)
User role assignment relationship	Verification state
Meeting initiator	Role (operation attribute at verification state)
Change time place operation	Requested action/event trigger when the associated guard if any is evaluated
Change time place operation	Requested states, state entry/do/exit actions
Meeting agreement	Context of class (from secure UML. resource) where state machine execution occurs
Pre condition: meeting initiator.user = ”Bob”	Guard condition on requested state transition of change time place

agreement details. These requirements are depicted in the secure UML model.

We construct the State Machine diagram by applying the mapping rules of secure UML to UML State Machines presented in the proposed approach on the following change time place () operation. The authorization constraint associated with change time place () operation is given below. Authorization Constraint with Operation Change Time Place (). Context Meeting Agreement::change Time Place (): void Pre: meeting Initiator. user = “Bob”). The following Table 2 presents the mapping of secure UML

model elements to UML State Machines for the Change Time Place protected operation of the meeting scheduler to Stat Machines.

When we apply the mapping rules and the steps for creation of state machines are followed than we construct the state machine diagram as presented in the following Fig. 3.

When we apply the test path generation technique by Ceneys *et al*. (2009), by following the syntax, we are able to generate the following test paths from the above given State Machine diagrams in Fig. 3:

- Change Time Place (time: Date Time, place: String) $ Meeting Agreement @ Requesting Change Time Place→ (meetingInitiaor. user = Bob) Process Update Schedule
- Change Time Place (time: Date Time, place: String) $ Meeting Agreement @ Requesting Change Time Place→ (meetingInitiator. user! = Bob) Request Rejected

CONCLUSION

Through our experimental results from various case studies, we found that Secure UML model can be used to design UML behavioral model (such as State Machine) by following predefined mapping rules. An additional advantage of the transformation to UML behavioral diagrams is that existing MBT techniques can be applied to model and test paths can be generated which can verify the behavior of the system under development.

Raimundas and Dumas (2011), have used transformation rules to map Secure UML model with UMLsec model but it required self assumption and modeller's knowledge about problem domain. Although their approach focus on transforming one secure model to another but ambiguities and incompleteness of transformation rules in different modeling situations hinders the abilities to properly specify RBAC policies in resulting models.

Secure UML model itself describes structural aspects of the system such as class members, operations and static assignment of authorization constraints on roles to perform particular operation. Through UML State Machine model we can present dynamic or behavioral aspects of system through operation execution and representation of states as a result of states transitions. Test paths generation using existing techniques enables us to verify the security policies and identify unsecure states of the system.

REFERENCES

Ceneys, A., A. Normantas and L. Radvilavicius, 2009. Designing role based access control policies with UML. J. Eng. Sci. Technol. Rev., 2 (1): 48-50.

Gray, M., 2004. Software security testing. J. IEEE Comput. Soc., 2(5): 32-36.

Jurjen, J., 2002. UMLsec: Extending UML for secure system development. Proceeding of the 5th International Conference on Unified Modeling Language, pp: 412-425.

Lindholm, J., 2006, M.A. Thesis, Department of Computer Science, University of Helsinki, 2006.

Lodderstedt, T., B. David and D. Jurgen, 2002. Secure UML: A UML basid modeling language for model-driven security. Proceeding of the 5th International Conference on Unified Modeling Language, pp: 426-441.

Mariscal, J., T. Doan, L. Michel, S. Demurjian and T. Ting, 2005. Role slices: A notation for RBAC permission assignment and enforcement. Proceeding of 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security.

Matulevicius, R. and M. Dumas, 2011. Towards model transformation between secure UML and UMLsec for role based access control. Proceeding of the 2011 Conference on Databases and Information Systems. Netherlands, pp: 339-352.

Raimundas, M. and M. Dumas, 2011. Towards model transformation between SecureUML and UMLsec for role-based access control. Proceedings of the 2011 Conference on Databases and Information Systems, pp: 339-352.

Shaukat, A., C.L. Briand, J. Rehman, H. Asghar, M.Z. Iqbal and A. Nadeem, 2006. A state-based approach to integration testing based on UML models. J. Inform. Software Technol., 49(11-12): 1087-1106.

Thapa, V., E. Song and H. Kim, 2010. An approach to verifying security and timing properties in UML models. Proceeding of the 15th IEEE International Conference on Engineering of Complex Computer Systems. Oxford, Mar. 22-26, pp: 193-202.

William, E.C., 2006. Software testing and the UML. Proceeding of 1st Workshop on Model-based Testing and Object Oriented Systems.