Tool support for code generation from a UMLsec property

How to cite:


For guidance on citations see FAQs.

© 2010 The Authors

Version: Accepted Manuscript

Link(s) to article on publisher’s website:
http://soft.vub.ac.be/ase2010/

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.
ABSTRACT
This demo presents a tool to generate code from verified Role-Based Access Control properties defined using UMLsec. It can either generate Java code, or generate Java code for the UML model and AspectJ code for enforcing said RBAC properties. Both approaches use the Java Authentication and Authorization Service (JAAS) to enforce access control.

Categories and Subject Descriptors: D.2.2 Software Engineering: Design Tools and Techniques [Computer-aided software engineering (CASE)]

General Terms: Security

1. INTRODUCTION
Security requirements can be made explicit on the design level, such as annotations on a UML model. UMLsec [4] extends UML to allow one to express security properties on a model, but it is still the developer’s responsibility to implement the code that will actually enforce those properties. This process can generate bugs and will not give any guarantee about how the implementation conforms to the model.

In this demo, we present a tool that generates Java and AspectJ code from a UML model with a verified UMLsec property. It can either generate only Java code, or, alternatively, implement the security property using AspectJ while still using Java for the functional code. The tool also has other features for UMLsec models verification that are not discussed here.

The next sections are organised as follows: we first give a short overview of UMLsec in section 2, then in section 3

*This work was partially supported by the EU project “Security Engineering for Lifelong Evolvable Systems (Secure Change)” (ICT-FET-231101)
†Part of this author’s work was done as a MSc student at the University of Namur, Belgium, under the supervision of Pierre-Yves Schobbens and Hubert Toussaint

Copyright is held by the author/owner(s).
ASE’10, September 20–24, 2010, Antwerp, Belgium
ACM 978-1-4503-0116-9/10/09.
This work focuses on a new feature: code generation from a UML model with an UMLsec RBAC property, as described on Figure 1. Given an UML model with UMLsec annotations describing an RBAC property, we generate code that conforms to the RBAC property. Two different approaches have been implemented, both using the JAAS [1] framework: the first one produces only Java code, while the second one uses AspectJ to enforce the RBAC property.

Both code generation techniques have been implemented using Aspect-Oriented Programming [3]. While code generation from the UML model is done in Java, we use AspectJ to add generation of code enforcing the UMLsec property, which has several advantages over an Object-Oriented only implementation. First, it allows us to clearly separate the code generation of the UML model itself from the code generation of the associated UMLsec property. Second, it allows us to easily extend the tool to generate code from other UMLsec properties by simply writing a new aspect, and reusing the existing Java code for generating to code corresponding to the UML model. And finally, it allows us to study the potential conflicts between several UMLsec properties in terms of an aspect composition problem.

3.1 Object-Oriented code generation

The first approach produces only Java code, both for implementing the UML model and the UMLsec property. The aspect responsible for generating code enforcing the RBAC property simply monitors the code generation process, and adds the necessary lines of code in the places where they are needed. The access control code is therefore spread all over the code base. While it might make modifications like adding (resp. removing) access control protection to a non-protected (resp. from a protected) method or attribute, it has the advantage of not requiring the use of Aspect-Oriented Programming and its drawbacks, like potential conflicts between aspects or negative impact on performances. Figure 2 shows an example of the lines of code added to protect a call to a myMethod method in a MyClass class, and then the lines of code added to call it.

3.2 Aspect-Oriented code generation

The second approach produces Java code for implementing the UML model, but the UMLsec property is implemented using AspectJ. Here, the RBAC code generation aspect monitors the code generation and simply creates an aspect while the functional code is being generated. This keeps the functional code separated from the authorisation code, making the modification of the code easier, and helping maintaining a traceability link between the access control code and the UMLsec property. Furthermore, any subsequent modification of the functional code will require little or no modification of the access control aspects. Figure 3 shows the same example as Figure 2, but using aspects.

Figure 2: Sample code added to protect a method and to call it

Figure 3: Sample code added to protect a method called myMethod to an access control aspect

4. RELATED WORK

There exist a lot of tools for generating Java (or other Object-Oriented language) code from UML models, like IBM's Rational Rose or ArgoUML. None of them, however, allows one to verify that a model enforces an UMLsec RBAC property, and generate code that conforms to it.

SecureUML [6] is an alternative to UMLsec for developing RBAC properties on a UML model. The authors developed a prototype implementation of their approach to translate RBAC properties expressed using SecureUML into EJB code. However, their approach does not include model-level verification as UMLsec does prior to the code generation, and they do not offer the opportunity to chose between several paradigms for the target code.

5. CONCLUSION AND FUTURE WORK

The tool we presented allows one to generate code from a RBAC property expressed in UMLsec in two different ways: by generating only Java code, or by generating Java code for the functional model as well as AspectJ code for enforcing the RBAC property. The generation of the code that enforces access control properties is implemented in AspectJ.

In future work, we will add support for code generation using other security frameworks than JAAS, like EJB. We will also work on code generation from other UMLsec properties. This will raise new and interesting challenges, as we will need to generate code that enforces several different properties, without introducing conflicts.

6. REFERENCES