Design Authorization Systems Using SecureUML

By Rudolph Araujo & Shanit Gupta, Foundstone Professional Services

February 2005
Overview

This whitepaper describes the Foundstone SecureUML template, a Microsoft Visio template built to model authorization systems. The tool allows architects to leverage the power and flexibility of the Visio environment while modeling their role-based access control systems. SecureUML is based on the widely known Unified Modeling Language (http://www.uml.org). It provides a high level of abstraction using visual notation and is therefore, well suited for architects and developers who may not have a strong security background.

Introduction

The need for security in large scale distributed systems has become a major priority for all organizations. The increasing number of software vulnerabilities has repeatedly shown that the design and engineering of these systems from a security stand point is often lacking. Security features are often added in an ad-hoc basis during the later part of the integration process resulting in errors and vulnerabilities that provide a potential for exploitation. One possible reason for this approach may be that security has only recently been considered an integral part of the software development lifecycle. Organizations are beginning to address this need by implementing changes in their development programs and by investing in security education for developers and software designers with classes like Foundstone’s Writing Secure Code classes.

Foundstone has been a leading software security advocate by offering consulting, training, and free tools to assist developers and designers adopt software security best practices. Through our many software reviews and penetration tests, we have seen first hand the implications of poorly designed software. Now, with the release of the SecureUML template, Foundstone aims to better equip designers with a tool to engineer more secure software.
The Need for SecureUML

There are several advantages to integrating security engineering in the software development lifecycle. To begin with, this approach allows security requirements to be integrated into system designs at a high level of abstraction. This further facilitates the development of security aware applications that avoid the violation of security policies. Moreover, by utilizing SecureUML to model the access control infrastructure can prevent errors during the implementation of access control policies and enables the technology independent development of secure systems.

The most commonly known architectural flaws are:

1. Incorrect use of cryptography
2. Incorrect user management techniques
3. Bad authorization design
4. Inefficient authentication mechanisms
5. Incorrect and inefficient data validation rule set

Implementation bugs are a byproduct of insecure design. In a race to fix bugs and security vulnerabilities the product developers are always being left behind. This reactive strategy often results in vulnerabilities that can be exploited by hackers. Many studies have shown that it is far less expensive to catch a bug in the design process than catching it after implementation.

The reasons why security policies are not integrated in the design phase is primarily due to:

1. Lack of knowledge – Until recently, many software architects have not fully recognized the need for secure software design.
2. Costs – Integrating the security policies and procedures increases production costs.

The ongoing discovery of vulnerabilities will continue to emphasize the need for secure software development. We believe that this trend has already begun and that organizations are starting to see that software security is a measure of application reliability that is at least as important if not more than performance.
Benefits of Using SecureUML

By leveraging the capabilities of Microsoft Visio, including the ability to integrate it with a team’s other UML diagrams and design documentation, the Foundstone SecureUML template provides security designers a clean and maintainable tool for documenting authorization models and decisions.

SecureUML helps developers by:

- Identifying poor authorization design and implementations
- Helping to find contradictions / holes like backdoors
- Identifying authorization bypass opportunities
- Encouraging the use of centralized authorization control
- Preventing the use of undocumented assumptions
- Being ideal for use with role based access control

Further, authorization design can take into account the threat model and can provide input to the threat model as well.
SecureUML Background

SecureUML is based on an extended model for role-based access control (RBAC). RBAC is a well established access control model with widely recognized advantages. RBAC lacks the ability to support expression of access control conditions that refer to the condition of a system. SecureUML therefore introduces the concept of authorization constraints. An authorization constraint is defined as a precondition for granting access to an operation.

SecureUML offers significant design extensibility because it combines the simplicity of graphical notation for RBAC with the power of logical constraints on models. Simple policies can be expressed using role-based permissions and more complicated requirements can be specified by adding authorization constraints with the resulting combination being quite powerful.

SecureUML is a modeling language that defines a vocabulary for annotating UML-based models with information relevant to access control. SecureUML defines a vocabulary for expressing different aspects of access control, like roles, role permissions, and user-role assignments. Due to its general access-control model and extensibility, SecureUML is well suited for business analysis as well as design models for different technologies.

SecureUML MetaModel

The SecureUML metamodel, is defined as an extension of the UML metamodel. The concepts of RBAC are represented directly as metamodel types.

SecureUML introduces the new metamodel types User, Role, and Permission as well as relations between these types. Protected resources are represented in a different way. Instead of defining a dedicated metamodel type to represent them, SecureUML allows every UML model element to take the role of a protected resource. Additionally, SecureUML introduces the type ResourceSet, which represents a user defined set of model elements used to define permissions or authorization constraints.

Permission is a relation object connecting a role to a ModelElement or a ResourceSet. The semantics of permission is defined by the ActionType elements used to classify the permission. Every Action-Type represents a class of security relevant operations on a particular type of protected resource. A method with the security relevant action execute or an attribute with the actions change and read are examples of this. In SecureUML, there is a corresponding action type for every class of such actions. Action types may also represent more conceptual classes of operations at a higher abstraction level.

---

The set of action types available in the language can be freely defined using ResourceType elements. A ResourceType defines all action types available for a particular metamodel type. The connection to the metamodel type is represented by the attribute baseClass, which holds the name of a type or a stereotype. The set of resource types and their action types, and the definition of their semantics on a particular platform, define the resource type model for the platform.

An AuthorizationConstraint is a part of the access control policy of an application. It expresses a precondition imposed on every call to an operation of a particular resource, which usually depends on the dynamic state of the resource, the current call, or the environment. AuthorizationConstraint is derived from the UML core type Constraint. Such a constraint is attached either directly or indirectly, via permission, to a particular model element representing a protected resource.

**SecureUML – Design Steps**

- Identify Users
- Identify application roles
- Map users into roles
- Identify resources
- Identify actions
- Identify authorization constraints
- Account for cardinality and complex relations – Inheritance
- Combine relevant diagrams to document security policy
Steps to Integrate SecureUML in Microsoft Visio

1. Make a note of the DIR path where SecureUML is installed. The default path is C:\program files\Foundstone Free Tools\Secure UML.
2. Open Visio and browse to "Tools\Options".
3. Select the "Advanced" tab.
4. Click the button labeled "File Paths..."
3. Add DIR to the directory lists for stencils
4. Add DIR to the directory lists for templates.
5. Restart Visio.
6. The SecureUML template will be present in Category “(Other)”

7. It can also be accessed from File->New-> Secure UML
8. Secure UML will now be listed in the shapes pane.
Example: E-Commerce Web Application Design Using SecureUML Template

Build an E-commerce application that allows users to browse, select products, and buy products online. The architecture diagram for an E-Commerce application would look like:

![Diagram of E-Commerce Web Application Design]

We will build the design for this generic e-commerce application, taking into account all the security policies using SecureUML principles and the tool provided.

Start a new drawing and choose SecureUML template using the integration steps mentioned in the previous section.
Step 1

The first step will be to identify all the users of the application and any hierarchy that might exist. The most common users for any application are:

- End Users
- E-Commerce Web Site Developers
- Sales Executives
- Web Administrators
- Database Administrators
Step 2

Identify the roles within the application making note of any hierarchies that may exist.

- End User
- Website Developer
- Product Manager
- Web Master
- Database Administrator
- Shipping Manager
Step 3

Assign the roles to specific users.

Step 4

Identify the resources that need to be protected.

- Product Information
- Customer Database as a whole
- Shipping Workflow
Step 5

Identify the possible operations on each resource.

- **Product Information**
  - Read
  - Write
  - Add / Delete Product

- **Customer Database as a whole**
  - Backup
  - Create / Delete
  - Administer

- **Shipping Workflow**
  - Place Order
  - Cancel Order
  - Review Information by Product ID

Step 6

Assign specific permissions to the roles identified in Step 2.

Note: The following is only one of a large number of solutions and makes significant assumptions such as the resource “Product Information” refers to information about products in all forms e.g. through the E-Commerce web site as well as in the Customer Database. As can be seen, this is far from being a trivial task and questions the assumptions designers and developers make.
About Foundstone Professional Services

Foundstone Professional Services, a division of McAfee, offers a unique combination of services and education to help organizations continuously and measurably protect the most important assets from the most critical threats. Through a strategic approach to security, Foundstone identifies, recommends, and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively.

Foundstone’s Secure Software Security Initiative (S3i™) services help organizations design and engineer secure software. By building in security throughout the Software Development Lifecycle, organizations can significantly reduce their risk of malicious attacks and minimize costly remediation efforts. Services include:

- Source Code Audits
- Software Design and Architecture Reviews
- Threat Modeling
- Web Application Penetration Testing
- Software Security Metrics and Measurement

For more information about Foundstone S3i services, go to [www.foundstone.com/s3i](http://www.foundstone.com/s3i).

Foundstone S3i training is designed to teach programmers and application developers how to build secure software and to write secure code. Classes include:

- [Building Secure Software](http://www.foundstone.com/s3i)
- [Writing Secure Code – ASP.NET (C#)](http://www.foundstone.com/s3i)
- [Ultimate Web Hacking](http://www.foundstone.com/s3i)

For the latest course schedule, go to [www.foundstone.com/education](http://www.foundstone.com/education).