Abstract

The Department of Defense (DoD) policy of multi-level security (MLS) has long employed the Bell-
LaPadula and Biba approaches for confidentiality and integrity; more recently, the Multiple Independent
Levels of Security/Safety (MILS) approach has been proposed. These approaches allow designers of
software-intensive systems to specify security levels and requirements for access to protected data, but
they do not enable them to predict runtime behavior. In this article, model-based engineering (MBE) and
architectural modeling are shown to be a platform for multi-dimensional, multi-fidelity analysis that is
conducive for use with Bell-Lapdula, Biba, and MILS approaches and enables a system designer to
exercise various architectural design options for confidentiality and data integrity prior to system
realization. In that way, MBE and architectural modeling can efficiently be used to validate security of
system architectures and thus gain confidence in the system design.

Security Challenges – An Introduction

A system designer faces several challenges when specifying security for distributed computing
environments or migrating systems to a new execution platform. Business stakeholders impose
constraints due to cost, time-to-market requirements, productivity impact, customer satisfaction concerns,
and the like. Thus, a system designer needs to understand requirements regarding the confidentiality and
integrity of protected resources (e.g., data). In addition, a designer needs to predict the effect that security
measures will have on other runtime quality attributes such as resource consumption, availability, and
real-time performance. After all, the resource costs associated with security can easily overload a system.
Nevertheless, security is often studied only in isolation and late in the process. Furthermore, the
unanticipated effects of design approaches or changes are discovered only late in the life cycle, when they
are much more expensive to resolve.¹

Model-Based Engineering for Security Analysis

Modeling of system quality attributes, including security, is often done—when it is done—with low
fidelity software models and disjointed architectural specifications by various engineers using their own
specialized notations. These models are typically not maintained or documented throughout the life cycle
and make it difficult to obtain a system view. However, a single-source architecture model of the system
that is annotated with analysis-specific information allows changes to the architecture to be reflected in
the various analysis models with little effort; those models can easily be regenerated from the architecture
model. This approach also allows the designer to conduct adequate tradeoff analysis and evaluate
architectural variations prior to system realization, gaining confidence in the architectural design. Models
also can be used to evaluate effects of reconfiguration and system revisions in post-development phases.
Using model-based engineering (MBE) tools, the SEI has developed analytical techniques to

1. represent standard security protocols for enforcing confidentiality and integrity, such as Bell-LaPadula [1] [2], Chinese wall [3] [4], role-based access control [5], Biba model [6]
2. model and validate security using system architecture according to flow-based approaches early and often in the life cycle

The MBE tools that the SEI uses are the Architecture Analysis & Design Language (AADL) and the Open Source AADL Tool Environment (OSATE) set of analysis plug-ins [7]. The AADL is used to model and document system architecture and provide a platform for analyses

1. using a single architecture model to evaluate multiple quality attributes, including security
2. early and often during system design or when upgrading existing system architecture
3. at different architecture refinement levels as information becomes available
4. along diverse architectural aspects such as behavior and throughput

**Architectural Considerations**

Security as an architectural concern crosses all levels of the system (application, middleware, operating systems, and hardware). Security requires thus intra- and inter-level validation and has immediate effects on the runtime behavior of the system, specifically on other dependability attributes.

The designer needs to enforce inter-level and intra-level security through the architecture. Fig. 2 depicts various system levels involved in the validation of security privileges against confidentiality requirements. (The example assumes that authentication and other necessary security services are enforced.) The designer seeks to ensure that the software applications do not compromise the confidentiality of the secure information they are exchanging. Consequentially, software applications need to execute on top of a secure operating system, be mapped to a protected and secured hardware memory space, and communicate over a secure communication channel. If the data is labeled confidential, then, every architectural layer needs have a clearance of at least that level.
In addition, the designer needs to acknowledge that security comes with a cost. Encryption, authentication, security, and protection mechanisms increase bandwidth demand in terms of CPU, network, and memory. These increases affect temporal behavior of the system (worst-case execution time, response time, schedulability, and end-to-end latency) as well as power consumption (especially important in battery-driven, or limited-life time devices such as sensor networks or portable communication devices).

As a result, security cannot be considered in isolation. The system designer makes choices to trade these quality attributes against each other. (This is a particular concern for embedded and real-time systems, which operate under significant resource constraints while ensuring high levels of dependability and security.) Security is interlinked with the other non-functional behaviors such as predictability/timeliness, resource consumption, and it inadvertently affects reliability and availability. Fig. 3 illustrates some of those dependencies on the single model-multiple analysis view.

An MBE Approach to Validating Confidentiality

Confidentiality addresses concerns that sensitive data should be disclosed to or accessed only by authorized users (i.e., enforcing prevention of unauthorized disclosure of information). Data integrity is closely related, as it concerns prevention of unauthorized modifications of data.
To model and validate the confidentiality of a system, we distinguish between general and application-dependent validation. General validation of confidentiality is the process of ensuring that a modeled system conforms to a set of common conditions that support system confidentiality independent of a specific reasoning framework for security. MBE takes advantage of the versatile concept of subjects operating on objects by permissible access (read, execute, append, and write), a notion introduced by Bell and LaPadula, enabling us to model and validate security at both the software and hardware levels.

This form of validation assumes that subjects and objects are assigned a security level that is the minimum representation to enforce basic confidentiality and need-to-know principles. By contrast, application-specific validation relies on detailed confidentiality requirements and a specific reasoning-based security framework.

The MBE security framework features

- representation of the confidentiality requirements of resources (i.e., objects)
- representation and generation of security clearance/(least) privileges of subjects operating on the objects
- representation of an access matrix, specifying allowed access operations of subjects on objects to support integrity

With the object’s security requirements specified in an AADL model, the least amount of privileges for the subjects can be generated in a straightforward manner. Given that the subjects’ privileges are specified, a mismatch between the least privilege and what has been specified means the assigned privilege is either insufficient or greater than the minimum privilege. The latter result may be unnecessary or an indication that the subject might be associated with objects not yet described in the model.

The following types of security validation and analysis are available as OSATE plug-ins:

- Basic confidentiality principle—access should only be granted if given the appropriate security clearance
- Need-to-know principle—access should be granted only to a resource if there is a need
- Controlled sanitization—lowering the security level of an object or subject should only be authorized and performed by a privileged subject
- Non-Alteration of Object’s Security Requirements—a subject using an object as input should not alter the security level of the object, even if the object is updated as an output from the subject.
- Hierarchical condition—(1) a component has a security level that is the maximum of the security levels of its subcomponents, and (2) all connections are checked to determine whether the source component of a connection declaration has a security level that is the same or lower than that of the destination component

Using OSATE and the AADL, system designers and developers can add analysis techniques as they need them.

The validation through architectural modeling of system security given confidentiality requirements of data objects and security clearance by users must include validation of

1. software architecture
2. system architecture where the software architecture is mapped to hardware components
By mapping the entities of a software architecture (e.g., processes, threads, and partitions) to a hardware architecture (consisting of, for example, CPUs, communication channels, and memory), we can ensure that the hardware architecture supports required security levels, as described in Figures 4 and 5.

---

```
-- Property intended to be customized by modelers.
-- Parameterizes the security property definitions.
property set Security_Types is
  -- Military levels by default
  Classifications:
  type enumeration (unclassified, confidential, secret,
  top_secret);

  -- This must be the first element of Classifications
  Default_Classification:
  constant Security_Types::Classifications =>
  unclassified;

  -- Default set of categories
  Categories:
  type enumeration {A, B, C, D};
end Security_Types;
```

**Fig. 4: Specification of security levels**

```
property set Security_Attributes is

  Class: inherit Security_Types::Classifications =>
  value(Security_Types::Default_Classification)
  applies to (data, subprogram, thread, thread group,
  process, memory, processor, bus, device,
  system, port, server subprogram,
  parameter, port group);

  Category: inherit list of Security_Types::Categories =>
  ()
  applies to (data, subprogram, thread, thread group,
  process, memory, processor, bus, device,
  system, port, server subprogram,
  parameter, port group);

-- ...
end Security_Attributes;
```

**Fig. 5: Architectural components to which security levels and requirements can be connected.**

Consider the scenario of two communicating processes, both requiring a high level of security, because the data objects require secret clearance. The system platform in this scenario consists of a set of CPUs with hardware support for various algorithms that encrypt messages before network transmission. By modeling the system, we can represent and validate that processes and threads (now considered to be objects) can be executed (access mode) on CPUs (subjects) with adequate encryption support. Furthermore, we can validate that CPUs (objects) communicate data (access modes of writing and reading) over appropriately secured communication channels (subjects). In a similar fashion, we can enforce design philosophies saying that only processes of the same security level are allowed to co-exist within the same CPU or partition or that they can write to a secured memory.

The combination of AADL and the OSATE security plug-in tool have been put into use in industry. Rockwell-Collins used the technology to enable the high-assurance handling of data from multiple sensors having varying levels of security, such as airborne imagery field programmable gate array (FPGA). Typically, a high-assurance processor is used to securely tag variable input. An FPGA is powerful and fast. It is deemed easier to develop applications on an FPGA, which also reduces the cost.
and time-to-market. Further, the FPGA can be reprogrammed at runtime (e.g., to fix bugs), which can lower maintenance-engineering costs. Because FPGA behavior is more complex, architecture-level definition and analysis are needed. To this end, Rockwell-Collins developed architectural models of the FPGA using AADL and used the OSA TE tool to validate security and demonstrate the high-assurance potential of FPGAs.

**Validating MILS Architectures with the MBE Approach**

The AADL and OSA TE tools can be used to validate the security of systems designed using the Multiple Independent Levels of Security/Safety (MILS) architecture approach. MILS uses two mechanisms to modularize—or to “divide and conquer”—in architecting secure systems: partitions and separation into layers. The MILS architecture isolates processes in partitions that define a collection of data objects, code, and system resources and can be evaluated separately. Each partition is divided into three layers, each of which is responsible for its own security domain and nothing else:

1. **Separation Kernel (SK)**—responsible for enforcing data isolation, control of information flow, periods processing, and damage limitation
2. **Middleware Service layer**
3. **Application layer**

Thus, MILS separates security mechanisms and concerns into the following components types, classified by how they process data:

- **SLS**—Single-Level Secure component: processes data at one security level
- **MSLS**—Multiple Single-Level Secure component: processes data at multiple levels, but maintains separations between classes of data
- **MLS**—Multi-Level Secure component: processes data at multiple levels simultaneously and transforms data from one level to another

The strength of a MILS architecture lies in its reductionist approach to decompose a system into components of abovementioned types that would be more manageable to certify. These components are also mapped to partitions (and as mentioned earlier, the MILS architecture approach builds on partitioning as one key concept to enforce damage limitation and separation of time and space).

An MBE approach is conducive to the validation concerns most critical to MILS, including:

- **Validating the structural rigidity of architecture, such as the enforcement of legal architectural refinement patterns of a security component into SLS, MSL, and MSLS types.** Given that a MILS architecture design and system is decomposed into security components that can be certified in isolation, the structural rigidity concerns the legal mappings and connections of the components. The decomposition into SLS, MSL, and MSLS types can be applied to components, connectors, and ports. Furthermore, each component can be divided into parts using the product, cascade, or feedback decomposition patterns [9] [10] [11] (details are omitted due to space constraints). For example, an MSLS component with $n$ security levels can be decomposed into $n$ distinct SLS components. Thus, confidence in validation of an architecture increases with the fidelity of the modeling. By using architectural model in AADL to capture the security types, and multiple architectural levels, MBE analysis be conducted to validate the correctness of the decompositions and mappings.

- **Architectural modeling and validation of assumptions underlying MILS.** Fundamental to enforcement of security in a MILS architecture is that the system supports partitioning, specifically damage limitation and separation in time and space. By partitioning the system, one minimizes the risk of illegal component interactions among components and protects components from the faulty behavior.
This can be realized in the system architecture by ensuring fault-containment and deploying security-cognizant memory allocation, so that MILS components and tasks reside in protected memory spaces—and do not co-reside in same memory space if they differ in security levels. Similarly, separation in time can be ensured through avoiding the interleaved execution of tasks with different security levels, realized in partition scheduling and validating execution behaviors. The AADL supports the modeling of partitions and virtual processors, and the virtual machine mechanism is recognized as a key concept for providing robustness through fault containment because it provides time and space partitioning to isolate application components and subsystems from affecting each other due to sharing of resources. This architecture pattern can be found in the ARINC 653 standard [12]. A single-source architectural model in AADL can thus be used validated the security requirement in an architectural context, specifically the MILS composition, and the architectural assumptions required.

- **Validating requirements specific to the NEAT characteristics and the communication system.** MILS requires that its SK and the trusted components of Middleware Services are implemented so that the security capabilities enforce what is commonly referred to as the NEAT characteristics: Non-bypassable—security functions cannot be circumvented; Evaluable—the size and complexity of the security functions allow them to be verified and evaluated; Always invoked—security functions are invoked each and every time without exception; Tamperproof—subversive code cannot alter the function of the security functions by exhausting resources, overrunning buffers, or other forms of making the security software fail.

The MBE approach allows designers to assure that software applications execute on top of a secure operating system, map to a protected and secured hardware memory space, and communicate over secure communication channels. It also enables the analysis of security measures early and throughout the development life cycle.

**Conclusions**

The objective of a secure system implies that security clearances are given conservatively (as opposed to generously). The MBE approach supports this objective through enabling analysis of the architectural model to derive the minimum security clearance on components. By providing mechanisms to ensure that sanitization is conducted within allowed boundaries, the MBE approach enables the system designer to analyze and trace more threatening security risks, as since sanitizing actions are permitted exemptions of security criteria and rules, and as such should be minimized in the system.

Security analysis using the MBE approach also supports

- evaluation of an architecture configuration with respect to impact on other non-functional attributes, such as increases in power consumption, bandwidth usage, and performance
- validation of architectural requirements necessary to enforce the MILS approach to containing faults through partitioning and separation in time and space
- reduction of the effort necessary for recertification in the event of architectural changes

Furthermore, validation of security can be conducted at multiple layers and different levels of fidelity, early and throughout the development life cycle.
References


Notes

1 In a study conducted by NIST, it was observed that 70% of all defects are introduced prior to implementation, (i.e., requirements and system and software design phases). Yet, only 3.5% of the defects were detected in these phases, while 50.5% of the faults were detected in the integration phase. The defect removal cost ranged from 5x to 30x relative to the cost of removing the defect in the phase of introduction (if it had been detected). Other sources are reporting similar estimates; while the numbers vary, the conclusions do not [13].

2 The AADL, an international industry standard, incorporates an XML/XMI exchange format to support model interchange and tool chaining. AADL also can be used (i) with UML state and process charts through its UML profile, (ii) to drill into root causes and develop quantitative analysis as a follow-up to the SEI Architecture Tradeoff Analysis Method®, and (iii) in conjunction with assurance cases, to support claims made about the safety, security, or reliability of a system. The freely available OSA TE includes analysis plug-ins for performance, resource consumption, security, and reliability. For more information on MBE and AADL, see www.sei.cmu.edu/pcs and www.aadl.info.

3 The principle of least privilege has been identified as important for meeting integrity objectives; it requires that a user (subject) be given no more privilege than necessary to perform a job. This principle includes identifying what the subject’s job requires and restricting the subject’s ability by granting the minimum set of privileges required.

4 MILS has been proposed as an approach to building secure systems [8] [9]. MILS is a joint research effort of academia, industry, and government, led by the United States Air Force Research Laboratory. The MILS approach is based on the notion of separating, and thus limiting the scope and reducing the complexity of, the security mechanisms.
Author Biography and Contact Information

**Jörgen Hansson, Ph.D.**, is a senior member of the technical staff and is the lead of the Performance-Critical Systems (PCS) Initiative at the Carnegie Mellon® Software Engineering Institute (SEI), and an associate professor of computer science at Linköping University, Sweden. The initiative focuses on developing, maturing and transitioning analysis-based assurance and model-based engineering tools and practices for predicting the dependability and performance of software systems. He is the author of two books and 80 technical papers published in conferences, journals, and books that cover embedded real-time systems, architectural design and validation, real-time data management data, and resource and QoS management. Hansson holds bachelors and masters degree in computer science from University of Skövde, Sweden, and a doctorate in computer science from Linköping University, Sweden.

SEI  
4500 Fifth Avenue  
Pittsburgh, PA 15213  
Phone: (412) 268-6733  
Fax: (412) 268-5758  
E-mail: hansson@sei.cmu.edu

**Peter H. Feiler, Ph.D.**, is a senior member of the technical staff in the PCS Initiative at the SEI. He has authored more than 80 articles in the areas of dependable real-time systems, architecture languages for embedded systems, and predictable system analysis & engineering. Feiler is the technical lead and author of the SAE AS-2C Architecture Analysis & Design Language (AADL) standard. Feiler holds a Vordiplom degree in Math/CS from Technical University Münich and a doctorate in computer science from Carnegie Mellon University.

SEI  
4500 Fifth Avenue  
Pittsburgh, PA 15213  
Phone: (412) 268-7790  
Fax: (412) 268-5758  
E-mail: phf@sei.cmu.edu

**John Morley** is a member of the operating staff at the SEI. He has reported on model-based engineering and service-oriented architecture for SEI publications, has more than 20 years experience in writing and editing scientific and technical materials, and holds a master of arts degree in English literature from Duquesne University.

SEI  
4500 Fifth Avenue  
Pittsburgh, PA 15213  
Phone: (412) 268-6599  
Fax: (412) 268-5758  
E-mail: jmorley@sei.cmu.edu