Embedded Systems Security Co-Design

Matthew Eby, Jan Werner, Gabor Karsai, Akos Ledeczi
Institute for Software Integrated Systems
Vanderbilt University, Nashville, TN 37235
{firstname.lastname}@vanderbilt.edu

Abstract

There is an ever increasing concern about security threats as embedded systems are moving towards networked applications. Model based approaches have proven to be effective techniques for embedded systems design. However, existing modeling tools were not designed to meet the current and future security challenges of networked embedded systems. In this paper, we propose a framework to incorporate security modeling into embedded system design. We've developed a security analysis tool that can easily integrate with existing tool chains to create co-design environments that addresses security, functionality and system architecture aspects of embedded systems concurrently.

1. Introduction

Model Integrated Computing (MIC) [1] is gaining wide recognition in the field of embedded software design. Models represent embedded software, its deployment platform and its interactions with the physical environment. Models facilitate formal analysis, verification, validation and generation of embedded systems [2]. Hence, this approach is superior to traditional manual software development process. Although, there is modeling tool support for analysis of functionality, performance, power consumption, safety, etc., currently available tools incorporate little if any support for security modeling. As a result, security is looked at only once the complete system has been built. At best, this approach of addressing security in the last stages of development is inefficient taking large amounts of effort to achieve only modest improvements in security.

Many times vulnerabilities are only discovered once they have been exploited. We advocate modeling environments that incorporate security into the early design phase of embedded systems. In many embedded applications system resources are scarce. Added overhead for security can have drastic effects on performance. An ideal embedded software development environment will allow the engineer to analyze security and performance tradeoffs based on the hardware platform the system will run on.

2. Background and Motivation

MIC can meet the challenges of designing secure embedded systems. A key advantage of the model based approach is the abstraction of the application domain. This abstraction is facilitated through the use of DSMLs. A DSML provides a system designer a set of concepts that are specifically tailored for a certain application domain. In our case, the domain is networked embedded real-time systems, such as process control systems, automotive, avionics and robotics systems. A DSML with the proper level of abstraction hides the inconsequential details of a system while allowing the engineer to shift focus to more important aspects. There are many examples of DSMLs developed for embedded system design in different domains [MILAN [4], SMOLES [5], AADL [4]]. By extending embedded system DSMLs, we can add tool support for security analysis, validation, verification and generation. These security tools will extend the large tool chains that already exist for embedded system design.

3. General Approach

We will demonstrate a process for integrating security analysis into existing tool chains to create a security co-design environment. The approach taken is to create a common DSML that is used to capture and analyze security properties of systems. The advantage of this approach is that the effort needed develop the security analysis tool is only spent once. Then this tool can be incorporated into existing embedded systems languages with minimal effort. By defining mappings from an embedded system DSML onto the security analysis DSML, we can analyze the security properties the embedded system. Figure 1 illustrates the process of defining mappings from one or more DSMLs onto a language supporting security analysis and feeding the analysis results back to the DSML.

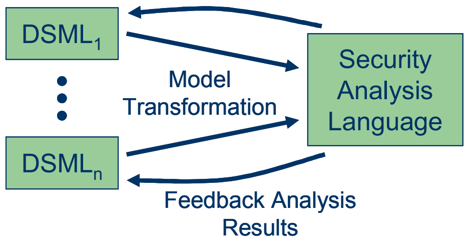

Figure 1. Mappings from DSMLs to SAL enable security analysis of the DSMLs

The co-design environment is implemented in the Generic Modeling Environment (GME) [2]. GME is a metaprogrammable tool which facilitates the graphical implementation of DSMLs through the use of metamodels. In this environment, we create a Security Analysis Language (SAL) that enables a user to model and analyze security related properties of embedded systems. (Note that while SAL is technically a DSML, from this point out we use the term DSML only in reference to a language for embedded systems design which we wish to add security analysis capabilities to.) The purpose of this analysis tool is to identify points in the system model that violate certain security requirements and provide useful feedback to the modeler. SAL allows such violations to be identified and remedied at design time before they can be exploited. Currently, SAL supports two types of analyses: information flow analysis and threat model analysis, which are detailed in the following sections.

3.1. Information Flow Analysis

The two traditional models for dealing with information flow in systems are the Bell-LaPadula model [6] and the Biba model [7]. Both of these models enforce an access control scheme that defines the rights of a subject to access information. Subjects and information are assigned a security level and a compartment which define what information a given subject is permitted to access. The set of all security levels is an ordered set that can be evaluated as an inequality (i.e. Top Secret > Secret). Compartments are a set that can be evaluated as an inequation (i.e. FBI ≠ CIA). The Bell-LaPadula model deals with confidentiality or secrecy of information in systems. The Biba model deals with integrity of information in systems.

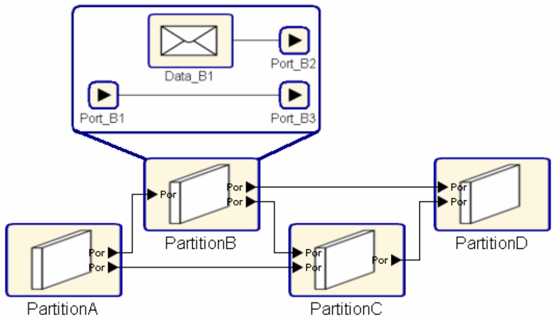

Figure 2. Partitions and dataflows in SAL

SAL views a system as a set of partitions, a set of data objects contained in each partition and the dataflows inside and across the partitions. Dataflows are represented as connections between input and output ports on a partition. In SAL, partitions are the subjects and are assigned a *security level* and *compartment* attributes. A data object inherits the *security level* and *compartment* classification of its containing partition. SAL allows the *security level* to be an integer value and the *compartment* to be a string value. Our analysis tool treats each data object as the root node in a tree search algorithm. The tool will traverse the dataflow paths originating from a data object and verify that each partition through which that data object flows has a *security level* and *compartment* that permit that partition to access the data object. Bell-LaPadula does not allow information to flow to a lower *security level* while Biba does not allow information to flow to a higher *security level*. Data objects in SAL are assigned two Boolean attributes, *secrecy* and *integrity*. The flow of every data object is evaluated based on the settings of these attributes. When *secrecy* is true the Bell-LaPadula model is enforced and when *integrity* is true the Biba model is enforced on the flow of that data object between partitions. Figure 2 shows a small example model in SAL.

3.2. Threat Model Analysis

The information flow analysis addresses potential security vulnerabilities in the logical channels explicitly defined for a system. In actual system these logical channels are implemented on a physical channel which is susceptible to attack. To prevent such attack, the communication channel can be encrypted. Adversary modeling in SAL enables the analysis tool to identify vulnerable channels and determine which encryption algorithms can be used to protect data being transmitted on that channel. Figure 3 illustrates the adversary model.

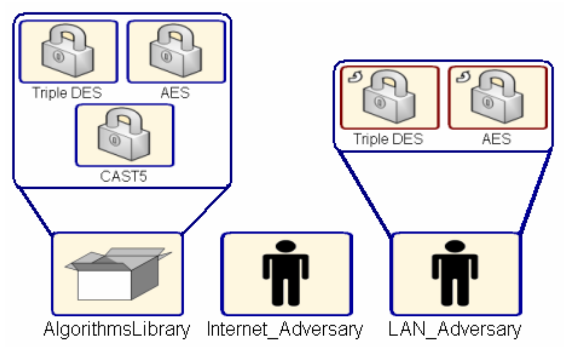

Figure 3. Encryption algorithms library and adversary models in SAL

In each system there is a library of encryption algorithms that contains the set of all encryption algorithms that can be used to encrypt a channel. Each system also contains a set of adversary models that define which encryption algorithms are vulnerable in the context of that adversary. Each adversary contains a set of references to algorithms that are defined in the algorithms library. Each reference has an attribute, *maxkeysize*, which means that the referenced algorithm is vulnerable to that adversary if the strength of its encryption is not greater than *maxkeysize*. Together, the encryption algorithm library and adversary models allow our analysis tool to determine which algorithms are safe to use to encrypt information flows. Each information flow in SAL has an attribute, *adversary*, which identifies the adversary model associated with that information flow. Each information flow in SAL also has an *EncryptionAlgorithm* and *KeySize* attribute. For each information flow in the system, the analysis tool checks the *EncryptionAlgorithm* and *KeySize* attribute against the set of encryption algorithms that are vulnerable for the adversary model specified by *adversary*.

3.3. Integrating Security Analysis with Existing Tool Chains

Although, there is modeling tool support for analysis of functionality, performance, power consumption, safety, etc., currently available tools incorporate little if any support for security modeling. As a result, security is only addressed once the complete system has been built. We want to leverage the work behind existing tool chains by incorporating security analysis in the system design process. SAL was created to be a reusable tool that can be integrated with multiple tool chains, thus reducing the effort that would be required to develop custom security analysis for each tool chain.

By defining a transformation that maps models of an embedded system DSML onto SAL, we can perform information flow analysis and threat model analysis on the embedded systems models. One of the powerful concepts of the MIC approach is easy composition of metamodels to form new languages. By composing the metamodel of a DSML with concepts from SAL, it is relatively easy to form these security specific extensions to an existing language. The tool designer can then create the transformation rules that map models in the DSML onto models in SAL.

Figure 4 shows a typical design flow for performing security analysis with an embedded system DSML.

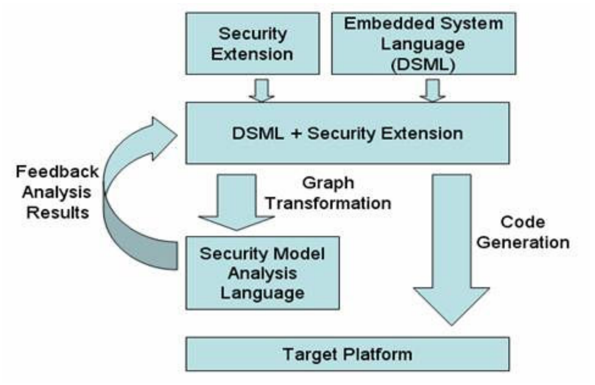
Security Extension
Embedded System Language (DSML)
DSML + Security Extension
Feedback Analysis Results
Graph Transformation
Code Generation
Security Model Analysis Language
Target Platform

Figure 4. Typical embedded system design flow with SAL

As a proof of concept, we have integrated SAL with an existing tool for the design of embedded systems called SMOLES [5]. For full description of the composition of SAL and SMOLES refer to [8].

4. Conclusion

We have demonstrated a security analysis tool that is capable of analyzing the flow of data objects through a system and identifying points in a distributed system that are vulnerable to attack. We have outlined a method for composing this type of security tool with existing tool chains for DSMLs. This approach leverages the development efforts that have gone into design of tool suites for existing embedded system DSMLs. Creating a separate analysis language for security properties allows reuse of this tool for multiple DSMLs.

7. Acknowledgement

This work was supported in part by TRUST (The Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422).

8. References

[1] Sztipanovits, J.; Karsai, G. *Model-integrated computing,* Computer Volume 30, Issue 4, April 1997 Page(s):110 – 111

[2] Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: *"Model-Integrated Development of Embedded Software,"* Proceedings of the IEEE, Vol. 91, No.1., pp. 145-164, January, 2003

[3] Kevin Poulsen, *Slammer worm crashed Ohio nuke plant network,* August 19 2003. Available at http://www.securityfocus.com/ news/6767

[4] Available from the Authors

[5] Szemethy, T. and Karsai, G. 2004. Platform modeling and model transformations for analysis. *Journal of Universal Computer Science 10*, 10, 1383–1406.

[6] D.E. Bell and L.J. LaPadula. "Secure Computer Systems: Mathematical Foundations and Model," Mitre Corp. Report No. M74-244, Bedford, Mass., 1975.

[7] K.J. Biba, "Integrity Considerations for Secure Computer Systems," Mitre Corp. Report TR-3153. Bedford. Mass., 1977.

[8] Eby, M., Werner, J., Karsai, G., Ledeczi, A., "Integrating Security Modeling into Embedded System Design." International Conference and Workshop on the Engineering of Computer Based Systems, IEEE, March, 2007