Formal methods in the security business: exotic flowers thriving in an expanding niche

David von Oheimb

Siemens Corporate Technology, Munich, Germany
David.von.Oheimb@siemens.com

Abstract. Formal methods in the industrial wild, outside the academic greenhouse, are still considered rather exotic, or even esoteric. Sometimes they are admired, more often smiled at, and most times simply ignored. There are some niches, though, where they display their abstract beauty. One of those places offering suitable environmental conditions is security. Which are the specific fertilizers there? Which particular sub-species have proven versatile and sturdy enough to survive in harsh industrial climate? Who recognizes the strong blessings of their hardly accessible blossoms? We share our grower's experience with them in the security field.

Keywords: Formal methods, security, software engineering, evaluation, models

Security as a software engineering problem

In the development of large IT systems, design errors and implementation bugs inevitably occur. Similarly to safety-critical systems, systems involving security-sensitive data face high risks because their failure can cause great damage. The risk involved with them is even higher than with safety-critical systems, because their deficiencies will not only cause problems accidentally, but will be searched for actively and exploited systematically.

One cannot expect to cope with the problem by legal or educational means, and organizational and physical measures have limited strength and scope. So the only really effective way to prevent attacks is by removing any potential loopholes and vulnerabilities. This is hard to achieve though: since security is a non-functional and holistic property that pervades the whole system and thus intricate to specify, and since systems are usually fairly complex, security flaws are notoriously hard to avoid, find, and correct. It's even harder to convince oneself, or one's contractors/customers, of the absence of such flaws.

The solution offered by formal methods

System security can only be approximated, by careful requirements analysis, systematic design and development, and extensive reviews and checks during all development phases. Formal methods provide the most rigorous tools for this.

During requirements analysis, abstract models help keeping the overview (by concentration on the essentials) and understanding the security issues and by a systematic approach, e.g. generic patterns simplifying the analysis.

During design and documentation, formalizations enhance the quality of specifications and other descriptions by preventing ambiguities, incompleteness, and inconsistencies.

During implementation, formal analysis can be used for systematically testing — or even verifying with mathematical precision — the correctness of the actual product wrt. its specification.

Although formal methods usually require high sophistication and large effort, in the security area the risks are often so high that it still pays to use them since they offer a higher level of assurance than any other known method. This explains why IT developers like Siemens are willing to apply them for their most security-critical products and solutions, and why standardized and widely used security evaluation criteria like the Common Criteria require their use for higher levels of assurance.

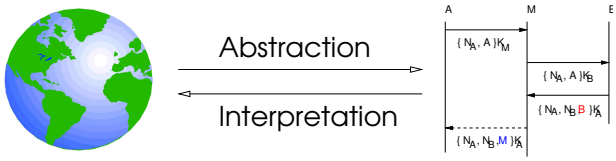
The use of security evaluation and certification is motivated in two ways:

intrinsically: developers can use it for internal quality control

extrinsically: developers are forced into it by market pressure or, more often, by legal requirements. According to our observations, in most cases this form of motivation is the decisive factor.

Formal security modeling

Every formal method naturally requires a formal model of the system to be analyzed. A *formal security model* is an abstract description (in an appropriate formal notation) of the real system and its desired properties, focusing on the relevant security issues.



The description includes the *security policy*, defining what actions, data flow, etc. is allowed, typically by a relationship between subjects and objects. The model usually describes also the mechanisms that are meant to enforce the given policy, typically in terms of system state or sequences of states. Security verification can then check (at the abstraction level of the model) whether the mechanisms actually enforce the policy. Even if the model is not used for subsequent testing or verification, the very act of producing a formal model is already of enormous value, since many errors already show up during this process.

Several factors influence the shape of a formal model.

The formality level should be adequate:

- the more formal, the more precision
- the less formal, the less special skills are required

The choice of formalism depends on:
- the application domain, modeler's experience, tool availability, ...
- its quality: it should be simple, expressive, flexible, and mature

The abstraction level should be:
- high enough to maintain overview and to minimize efforts
- low enough not to loose important detail

The use of refinement promises to offer the best of both high-level and low-level descriptions, yet at the cost of some extra effort

When formalizing a system, information on the following is required.

System architecture: which components exist and how they are connected

Security-related concepts: e.g., actors, objects, states, messages, ...

Threats, security goals, and objectives: describing which attacks shall be countered, e.g. for integrity: which data contents are only allowed to be modified by whom during which times, or on transit from where to where

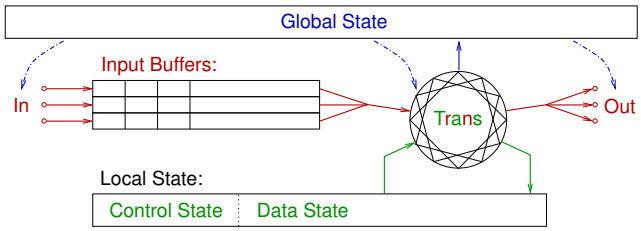
Security mechanisms: their relation to the goals and how they are applied, e.g. who signs which contents for what purpose and where signatures are checked. They should be described precisely but at high level, e.g., abstract message format/contents but not concrete syntax.

There are four classes of practically relevant formal security models, about which we briefly share our experience.

Automata models

The most general way of describing systems at an abstraction level suitable for security analysis is by state transition automata. Many such formalisms exist, e.g. the Input/Output Automata by Lynch and Tuttle [LT89]. Such automata can be seen as the basic model upon which the more specific classes of models, described below, are built.

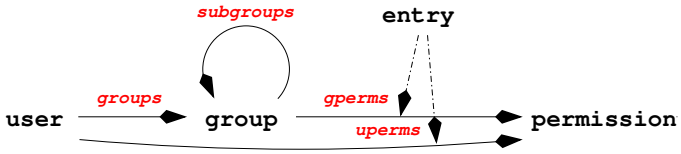
For convenient description of a large variety of reactive systems, we have introduced *Interacting State Machines (ISMs)* [OL04], whose distinctive feature is buffered I/O that can occur simultaneously on multiple connections. ISM models



can be verified with the interactive theorem prover Isabelle [NPW02]. We have applied ISMs when updating the Lotz-Kessler-Walter model [LKW99] employed as the formal security model of the Infineon SLE 66 smart card processor, which — with the help of the LKW model — was the first of its kind to receive an EAL5 certification. An ISM model [OLW05] was employed also for the EAL5 certification of the memory management of the successor chip, the SLE 88.

Access control models

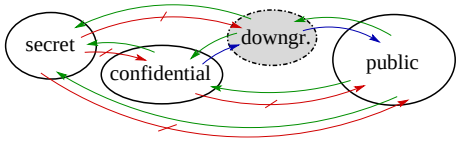
Classical access control models, like the well-known Bell-LaPadula model [BL73], have proven too restricted for practical use. Currently much more important are *role-based access control (RBAC)* models [SCFY96]. Subjects are related with roles, which may be hierarchically structured, and finally roles are related with access rights to objects.



We have used an RBAC model to describe the complex access control policies of a medical information system. It involves two independent hierarchies, one of roles and sub-roles determining privileges to perform certain actions, and one of groups and subgroups determining the permissions to access certain data elements. When we modeled the system according to the specifications provided by our customers, we asked them many "nagging questions", which lead to clarifications that boosted the quality of the specifications.

Information flow models

Classical information flow models include the noninterference model by Goguen and Meseguer [GM82] and many others, and prominent recent examples include [Man03]. They describe which information may flow between which domains in a very abstract way such that they can capture also indirect and partial flow of information.



Although we have contributed to this research field ourselves [Ohe04], we consider such models on the one hand too powerful and on the other hand too difficult to be of much practical use these days.

Crypto protocol models

Probably the most successful class of security models so far are crypto protocol models describing the message traffic of security protocols. Mostly secrecy and authentication goals can be specified and then verified automatically using model-checkers tailored for this application.

We have participated in a recent EU-funded project called *Automated Validation of Internet Security Protocols and Applications (AVISPA)* [AH-03], which dealt with the subject very successfully. One of its highlight applications was the H.530 authentication for mobile roaming in a multimedia scenario. Two vulnerabilities were found and corrected, and the solution is being patented.

References

AH-03. The AVISPA project homepage, 2003. http://www.avispa-project.org/.

BL73. D.E. Bell and L. LaPadula. Secure Computer Systems: Mathematical Foundations (NTIS AD-770 768), A Mathematical Model (NTIS AD-771 543), A Refinement of the Mathematical Model (NTIS AD-780 528). Technical Report MTR 2547, Mitre Corporation, Bedford MA, 1973.

GM82. J. A. Goguen and J. Meseguer. Security policies and security models. In *Symposium on Security and Privacy*. IEEE Computer Society Press, 1982.

LKW99. Volkmar Lotz, Volker Kessler, and Georg Walter. A Formal Security Model for Microprocessor Hardware. In *Proc. of FM'99 World Congress on Formal Methods*, volume 1708 of *LNCS*, pages 718–737. Springer-Verlag, 1999.

LT89. Nancy Lynch and Mark Tuttle. An Introduction to Input/Output Automata. *CWI Quarterly*, 2(3):219–246, 1989. http://theory.lcs.mit.edu/tds/papers/Lynch/CWI89.html.

Man03. H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security*. PhD thesis, Univ. d. Saarlandes, 2003.

NPW02. Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. See also http://isabelle.in.tum.de/docs.html.

Ohe04. David von Oheimb. Information flow control revisited: Noninfluence = Noninterference + Nonleakage. In P. Samarati, P. Ryan, D. Gollmann, and R. Molva, editors, *Computer Security – ESORICS 2004*, volume 3193 of *LNCS*, pages 225–243. Springer, 2004. http://ddvo.net/papers/Noninfluence.html.

OL04. David von Oheimb and Volkmar Lotz. Formal Security Analysis with Interacting State Machines. In Gerwin Klein, editor, *Proc. NICTA Formal Methods Workshop on Operating Systems Verification*, pages 37–72, Sydney, Australia, 2004. National ICT Australia, Technical Report 0401005T-1. http://ddvo.net/papers/FSA_ISM.html.

OLW05. David von Oheimb, Volkmar Lotz, and Georg Walter. Analyzing SLE 88 memory management security using Interacting State Machines. *International Journal of Information Security*, 4(3):155–171, 2005. http://ddvo.net/papers/SLE88_MM.html.

SCFY96. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.