Formal Methods and Access Control

Michael Huth, Imperial College London

Related Concepts

- Access-control architectures
- Formal verification
- Identity management
- Model-based development
- Policy languages
- Security policies

Definition

A formal method is any technique or method that aids in the construction and validation of computer-based systems and is based, in total or in part, on rigorous mathematics. Access control [8] refers to any method or mechanism by which the access of principals to resources is regulated; formal methods can aid considerably in the design, validation, and implementation of access control.

Background

Access control within IT systems developed historically through simple mechanisms such as access-control matrices [8]. Such a matrix explicitly lists allowed access, e.g., which principal can do what actions to which objects. This approach worked well for isolated or small-scale IT systems, but is ill-suited for systems that are distributed, federated, large-scale, or complex for other reasons.

Theory and Application

To address these needs, policies emerged as a standard tool for specifying and deciding access requests. Policies have expressive mechanisms that allow for more efficient representations of requests (e.g. through roles [5]), priority composition of rules (e.g. to define and enforce overriding concerns), the inheritance of privileges (e.g. through role hierarchies [5]), etc. A policy may be phrased in a manner that is machine-readable, appreciable by system users or both.

Policies are meant to capture intended behavior of an access-control system. Mathematically, we may think of a policy as a function $p\colon \mathsf{Req} \to \mathsf{Dec}$ where Req is the set of access requests and Dec is the set of decisions made for such access requests. Two such decisions are grant (permitted access) and deny (prohibited access). But the application domain or the composition of sub-policies may require additional values in Dec. We mention $\mathsf{conflict}$ (rules of sub-policies provide conflicting evidence for deciding an access request), gap (the policy provides no evidence for deciding an access request), error (evaluation of the policy for an access request causes some error), halt (to stop the execution of a program), and $\mathsf{irrelevant}$ (to express that the policy does not apply to an access request).

The decision of composed policies should be the composition of their individual decisions. For one, this makes policy behavior independent from syntactic peculiarities. For another, distributed systems may not allow the explicit construction of a composed policy. Formal methods can then help a great deal in developing techniques for combining local policy decisions into global ones, e.g. through research on policy algebras (e.g. [2]).

Policies within IT systems are not given as mathematical functions as above, but as syntactic, perhaps distributed, objects such as a rule file that specifies a firewall policy. Policy analysis (e.g. the one in [6]) is therefore an important formal activity. Such analyses include conflict detection [3] (e.g. are there conflicting rules in a policy?), gap detection [3] (is the policy not defined for some request?), safety problem (e.g. can access privileges be confined to intended owners of such rights?), and refinement [3] (e.g. is the modified policy less permissive than the original one?).

The policy decision point of an access-control system may also have to conduct policy analysis in order to determine how a policy ought to decide on a request. Consider delegation [1], in which a principal can transfer his or her right to, or share it with, other principals (e.g. for document collaboration). A policy decision point then needs to analyse the policy and its delegation statements in order to determine if granting an access request for a principle follows directly, or is derivable through a chain of policy-compliant delegations.

Policies are expressed in policy languages (e.g. compliant with the standard XACML) or, indirectly, in other mathematical formalisms (e.g. in type inference systems for the enforcement of secure information flow in Java programs). The methods used in policy analysis vary greatly but depend on said concrete representation of policies. Secure information flow of a Java program might be established by showing that one can infer its type to have a policy-compliant value in a security lattice. The presence of conflict in a rule-based policy (e.g. for a firewall) might be established by expressing such presence in terms of a propositional constraint and then checking the satisfiability of that constraint with a SAT solver. See [6] for use of a SAT solver in this context. In contrast, role engineering [5] might use purely probabilistic methods, as familiar from artificial intelligence, to discover functional roles (e.g. ones that don't already exist as such within an organization but that reflect common access privileges) into which principals can then be abstracted.

Formal methods can therefore serve several important but distinct functions in the design, verification, and implementation of access-control systems. Role discovery, e.g., can be considered as a design activity; proving secure information flow in programs can be seen as validation of access rights where principals are programming entities such as computation threads or applets on smart cards; and expressing rule-based policies as propositional constraints may lead to the discovery of redundancies in policies and so improve their implementation.

Applications of formal methods also concern the administrative aspect of policy-based access control [7]. For example, a formal policy language may cleanly separate between normal access privileges (e.g. permission to collabo-

rate on a document), administrative privileges (e.g. permission to invite others as collaborators on a document), and "super"-administrative privileges (e.g. permission to transfer ownership of a document). Engineering workable policies with such layered, administrative permissions is difficult to achieve without the aid of formal methods, which can support analysis and can provide robust administration patterns within a policy language.

Open problems

Access control in IT systems has to be done such that people can actually use those systems. Usability [4] in the security domain studies how the use of systems and their security interact. Future formal methods may help in improving the usability of access-control systems so that private citizens and non-specialist workers alike can control the access to resources for which they are responsible.

Formal methods are also believed to be needed in the realization of robust access control in tomorrow's systems that increasingly rely on virtualization, software as a service, and on cloud computing.

Recommended Readings

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin, A calculus for access control in distributed systems, ACM Transactions on Programming Languages and Systems, vol. 15, no 4 (1993), pp. 706–734.

[2] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, An algebra for fine-grained integration of XACML policies, Fifteenth Symposium on Access Control Models and Technologies, ACM, pp. 63–72, 2009.

[3] G. Bruns, D. S. Dantas, and M. Huth, A simple and expressive semantic framework for policy composition in access control. Proceedings of the Fifth Workshop on Formal Methods in Security Engineering: From Specifications to Code, ACM, pp. 12–21, 2007.

[4] Cranor L. F. and Garfinkel S., Security and Usability, O'Reilly Media, Cambridge, MA, 2005,

[5] Ferraiolo D. F., Kuhn D. R., and Chandramouli R., Role-Based Access Control (second edition), Artech House, Boston, MA, 2007.

[6] A. Jeffrey and T. Samak, Model Checking Firewall Policy Configurations, Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks, IEEE, pp. 60–67, 2009.

[7] N. Li and Z. Mao, Administration in Role-Based Access Control, Proceedings of the Second ACM Symposium on Information, Computer and Communications Security, ACM, 2007, pp. 127–138.

[8] Messaoud B., Access Control Systems, Springer, New York, NY, 2006.