Verifying Workflow Processes against Organization Security Policies

Carlos Ribeiro
Carlos.Ribeiro@inesc.pt

Paulo Guedes
Paulo.Guedes@inesc.pt

IST/INESC Portugal

Workflow applications for large complex organizations often need to cross several security domains, each with different management and specific security requirements. The resultant cross-dependency between the workflow specification and the security policy of each domain can be hard to manage without specific tools.

This work presents a static analyzer that automatically verifies the consistency between workflow specifications written in WPDL (Workflow Process Definition Language) and organization security policies, written in a security language specially designed to express simultaneously several security policies.

1 Motivation

On designing workflow processes for an organization it is essential to ensure that the enactment of the workflow specification does not violate the security policy of the organization. On small organizations the security violations caused by inconsistent specifications can be avoided by careful inspection of both the security policy and workflow specifications. However, on large and complex organizations even simple policies like preventing confidential documents to be sent outside the organization, or approvals to be performed by employees without authorization can be difficult to ensure without specific tools. Moreover, is not enough to verify each task individually, because a workflow process may enforce a security violation by specifying two tasks, which are individually allowed, but together violate a separation of duty policy.

Previous work on this subject [3, 2] used specialized workflow infrastructures to maintain both the workflow specification and the security policy, and developed specialized analyzers to verify the consistency of the jointly specification. This approach has some problems. First, it is difficult to extend the workflow infrastructure to cope simultaneously with several complex security policies like *information flow, chinese wall* and other forms of separation of duty. Second, workflow infrastructures are not usually designed to be used by symbolic solvers. Third it ties the security policy to one single workflow infrastructure, disregarding other systems of the organization.

Our approach is to use a generic security infrastructure, which is flexible enough to express simultaneously several complex security policies and satisfy the requirements of several systems and domains, hence be shareable by most systems across an organization.

The contribution of this paper is to show how a security specification written in a generic and flexible security language (SPL) can be checked for inconsistencies with a workflow specification written in an off-the-shelf workflow process definition language, namely WPDL (Workflow Process Definition language) [8].

2 Security language (SPL)

SPL [6] is a security language designed to express policies that aim to decide about the acceptability of events. The acceptability of each event depends on the properties of the event (e.g. author, target and action), on the context at that time and on the properties of past and future events. SPL entities are typed entities with an explicit interface by which their properties can be queried. Some of the entities manipulated by SPL are internal to SPL, like rules and policies, but most are external, like users, files, actions, objects and events. The properties of each external entity depend heavily on the platform (operating system, workflow engine) that implements those entities.

A SPL policy is composed of sets and rules. Sets contain the entities used by the policies to decide on event acceptability. Rules are functions of events, which may assume three values: *deny*, *allow* and *not apply*. Rules can be simple or composed. Simple rules are comprised of two binary expressions, one to decide on the domain of applicability of the rule and another to decide on the acceptability of the events to which the rule is applicable. Rules can be composed of other rules and policies through the use of simple ternary algebra. Each policy has one special rule called the "query rule" (identified by question mark) which is usually a composition of other rules and defines the behavior of the policy. Delegation is achieved when a policy is instantiated and used as a rule in the composition of other rules.

3 WPDL/SPL consistency verifier

Workflow processes have been expressed in many different models. Most of them use specialized workflow languages with specific flow constructions [8], some are based in colored annotated Petri nets [4], and a few others define processes as sets of constraints [7]. Although most of these models are not equivalent due to properties present in some of them and absent in others, it is often possible to create an equivalent model based on a different paradigm.

SPL defines a constraint model with almost all the restriction operations necessary to express a workflow process. However, the SPL constraint model was designed to be implemented by an event monitor and it is not suited to be used by a symbolic constraint solver, or to express task dependencies. Hence, it is necessary to define an intermediate model that is able to express the restrictions of both SPL and workflow systems, and is suitable to be used by a symbolic solver.

The proposed system is composed of two language translators to translate WPDL and SPL to a common constraint language, and a consistency verifier engine, to handle the constraints of both languages.

The consistency verifier engine is a symbolic constraint solver that knows how to handle each type of constraint in order to detect conflicts. It is composed of a set of rules comprising the knowledge on constraint resolution, and an engine that knows how to handle those rules. Rules are expressed in the CHR (Constraint Handling Rules) language [5], for which several engines already exist.

The engine works by repeatedly applying the CHR rules on the constraints produced by the language translators until a inconsistency is detected or no more simplification is possible. An inconsistency is detected when the engine simplifies one or more constraints into a *fail* constraint. Figure 1 it is shows some simple CHR rules to handle *less* and *less or equal* constraints. The rules with a "$\Leftrightarrow$" sign replace constraints on the left by constraints in the right. The rules with a "$\Rightarrow$" sign infer the constraints in the right from the constraints in the left.

```
X ≤ X              ⇔ true          // Reflexivity
X ≤ Y, Y ≤ X       ⇔ X = Y         // Antisymmetry
X ≤ Y, Y ≤ Z       ⇒ X ≤ Z         // Transitivity
X < X              ⇔ fail          // Ireflexivity
X < Y, X ≤ Y       ⇔ X < Y         // Subsumption
```

Figure 1. Some rules to handle ($<$) and ($\leq$) constraints.

4 Current status

Currently the consistency verifier engine is composed of 229 CHRs running over the CHR solver provided with SICStus Prolog [1], which are able to handle all SPL and WPDL constraints. Some experiences were performed using compositions of SPL policies and workflow specifications to validate the process. Namely, using simple access control policies, information flow policies, separation of duty policies, workflow specifications with loops, and policies with constraints over cardinality of role membership.

5 Conclusion

We have presented a tool that is able to statically verify the consistency between a workflow specification written in a standard workflow specification language, and a security policy written in a flexible security language capable of globally express the policy of complex organizations. To handle the complexity of the restrictions of both systems, the tool translates both security and workflow specifications into a common constraint language that can be symbolically checked using CHRs.

References

[1] J. Andersson, S. Andersson, K. Boortz, M. Carlsson, H. Nilsson, T. Sjöland, and J. Widn. SICStus prolog user's manual. Technical Report T93-01, Swedish Institute of Computer Science, Oct. 1995.

[2] V. Atluri and W.-K. Huang. An authorization model for workflows. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Proceedings of the Fourth ESORICS*, LNCS, pages 44–64, Rome, Italy, Sept. 1996. SV.

[3] E. Bertino, E. Ferrari, and V. Alturi. A flexible model for the specification and enforcement of role-based authorizations in workfiow management systems. In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control (RBAC-97)*, pages 1–12, New York, Nov. 6–7 1997. ACM Press.

[4] C. Ellis and G. Nutt. Modelling and Enactment of Workflow Systems. In M. A. Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1993.

[5] T. Frühwirth. Theory and practice of constraint handling rules, special issue on constraint logic programming. *Journal of Logic Programming*, pages 95–138, Oct. 1998.

[6] C. Ribeiro and P. Guedes. Spl: An access control language for security policies with complex constraints. Technical Report RT/0001/99, INESC, Jan. 1999.

[7] A. Tate. Representing plans as a set of constraints - the <I-NOVA> model. In B. Drabble, editor, *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 221–228. AAAI Press, 1996.

[8] WFMC. Interface 1: Process Definition Interchange Process Model (WFMC-TC-1016-P). Technical report, Workflow Management Coalition, Brussels, Nov. 1998.