

Runtime Prediction of Policy Violations in Automated Business Processes

(Extended Abstract)

Maike Gilliot and Rafael Accorsi

Department of Telematics
University of Freiburg, Germany
{gilliot|accorsi}@iig.uni-freiburg.de

Abstract. While obligations are essential to capture privacy and usage policies, they cannot be enforced by traditional reference monitors. We present a monitor architecture to anticipate policy violations based on Runtime Verification and on statistical knowledge about the outcome of previous executions. By monitoring the progress of processes, in particular their access decisions, the goal is to decide whether an access request potentially leads to a violation of pending obligation.

1 Problem: Monitoring Obligations

Impelled by scandals and corporate misbehavior, compliance frameworks, such as HIPAA, SOX and Basel II, have been introduced to prescribe (e.g. privacy and usage) requirements for the realization of business processes (or simply *processes*). Here one needs to ensure that processes adhere to the corresponding regulatory requirements (*policies*) of the corresponding compliance framework.

To this end, compliance requires policies to be enforced to *prevent* policy violations. The traditional approach to do so builds on the concept of inline reference monitoring, i.e. monitors that observe the steps of the process execution and halt it if the next step violates a policy. Formally, inline monitoring effectively enforces the so-called *safety* properties [5,7].

However, according to Breaux et al. [3] around 70% of the policies are *co-safety* properties whose violations cannot be prevented by means of execution monitors. This happens because the rules in these policies include *obligations*, i.e. actions a subject must perform as a consequence of a granted access rights, such as the deletion of accessed data or notification of access. Since execution monitors can only observe, albeit not anticipate, whether obligations are eventually fulfilled or not, prevention of violations is technically infeasible. The following excerpt of the Patriot Act (31, Section 103.122) exemplifies the situation:

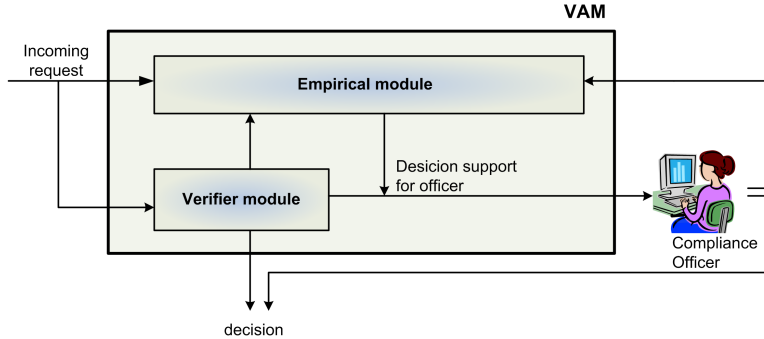


Fig. 1. VAM architecture

“The customer identification procedure and transaction reporting for acquisition of bonds, stock exchanges including hedge funds over US\$ 15K must include risk-based procedures for verifying the identity of each customer [...]. The identification and reporting procedures must be timely [...].”

In this rule, the right to be granted, i.e. *authorization*, is the acquisition of some financial product, where the total amount denotes *provision* for this rule to be triggered. The *obligation* prescribes that the acquisition must be followed by an identity check and corresponding reporting, which must occur in a timely manner. Taken as a policy rule, the monitor can intervene and forbid the acquisition of a certain product whenever the provisions or the authorization are not met. However, once the acquisition has been triggered, the monitor cannot enforce the identity check and reporting, it can only recognize that it has not been performed “timely.”

2 Approach: Anticipation of Violations

This paper presents the inline monitor architecture VAM for *a priori*, runtime anticipation obligation violations. VAM stands for “Violation Anticipation Monitor” and is schematically shown in Fig. 1.

Building on Accorsi et al. [2], whenever a subject requests the right to perform some activity during the process run, the monitor automatically checks whether this activity leads to a violation of obligations later in the run. In contrast to [2], which uses risk to anticipate violations, our approach is based on *runtime verification* [6] and statistical reasoning. To anticipate the outcome of a process with regard to policy adherence, the **Verifier module** of the VAM applies runtime verification – a lightweight

model-checking technique based on linear temporal logic (LTL) [4] – to compute the possible suffixes of the execution, answering in a 4-valued logic with “*true*,” “*false*,” “*presumably true*,” or “*presumably false*.” For the last two values, interactive decisions could be taken by a compliance officer as whether to grant or deny the right. Or, alternatively, whether to halt or proceed process’s execution.

Besides delivering formally founded decision support towards the future, our approach also considers the outcome of past decisions, using the **Empirical module** to this end. Specifically, based on a secure log file [1, §3], it includes information correlating the previous decisions taken, their outcome, and the current recommendation suggested by runtime verification. In doing so, it may happen that despite some obligation evaluates to “*presumably false*,” previous decisions demonstrate that the process eventually terminates in a state that satisfies the pending obligation. Such a feature is useful in several situations as a means to relativize the decision and tailor it for the current context.

References

1. R. Accorsi. *Automated Counterexample-Driven Audits of Authentic System Records*. PhD thesis, University of Freiburg, 2008.
2. R. Accorsi, Y. Sato, and S. Kai. Compliance monitor for early warning risk determination. *Wirtschaftsinformatik*, 50(5):375–382, October 2008.
3. T. Breaux and A. Antón. Analyzing regulatory rules for privacy and security requirements. *IEEE Transactions on Software Engineering*, 34(1):5–20, January/February 2008.
4. E. A. Emerson. *Handbook of Theoretical Computer Science*, chapter Temporal and modal logic. MIT Press, 1990.
5. L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, 3(2):125–143, March 1977.
6. M. Leucker and C. Schallhart. A Brief Account of Runtime Verification. *Journal of Logic and Algebraic Programming*, 2008.
7. F. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, February 2000.