A Practical Formal Model for Safety Analysis in Capability-Based Systems

Fred Spiessens and Peter Van Roy

Université catholique de Louvain,
Louvain-la-Neuve, Belgium
{fsp, pvr}@info.ucl.ac.be

Abstract. We present a formal system that models programmable abstractions for access control. Composite abstractions and patterns of arbitrary complexity are modeled as a configuration of communicating subjects. The subjects in the model can express behavior that corresponds to how information and authority are propagated in capability systems.

The formalism is designed to be useful for analyzing how information and authority are confined in arbitrary configurations, but it will also be useful in the reverse sense, to calculate the necessary restrictions in a subject’s behavior when a global confinement policy is given.

We introduce a subclass of these systems we call ”saturated”, that can provide safe and tractable approximations for the safety properties in arbitrary configurations of collaborating entities.

1 Introduction

Since Harrisson, Ruzzo, and Ullman (HRU) showed in 1976 [HRU76] that safety properties are generally intractable, two approaches have been explored to calculate a safe approximation for safety properties. The first one is to keep on using Turing Complete models and to deal with the intractability by limiting the resources allocated to the safety checker. The checker will “give up” after exhausting the given resources, and report the possibility of a safety breach without proof. Such an approach can for instance be implemented in the SPIN model checker [Hol97]. This allows the user of the model checker to iteratively increase the precision (depth) of the calculation.

A second approach builds tractability into the model: instead of calculating a finite approximation of a possibly intractable safety property, it tries to calculate the exact value of the corresponding tractable safety property in an approximate model. Take-Grant systems [BS79] are an example of this approach, in which the safety properties are tractable [LS77, FB96]. This is the approach we take in this paper. Because checking tractable models can take arbitrary many resources too, we will take care that the approximation can be easily adapted: coarsening the model in some regions to make it simpler while refining it in other regions to gain precision.

Regardless of the approach taken, model checking involves the translation from a real world situation to a configuration in the formalism, and from the
calculated safety properties to conclusions that can be applied to the actual problem. Both translations should be well understood by the user of the model checker and should be explicit and well documented.

To ensure that the formalism is practical and useful to software engineers, we aimed for these translations to be easily described in terms of programming and design properties. We want our formalism to be useful at all levels of abstraction, during all stages of the software building process. The precision of modeling can be iteratively adapted. The resulting formal system forms a suitable base for the implementation of a dedicated model checker.

We developed Authority Reduction Systems via a series of consecutive refinements starting from Take-Grant systems [BS79]. The structure of this paper coarsely reflects this history. We first give an introduction to capability based security in Section 2. As a running example, we describe in Section 3 a simple pattern of authority delegation and revocation, called the Caretaker [MS03]. We will use this pattern as a touch stone for the expressive power of our formalism.

From studying capabilities [DH65] in general, and especially from the clarifications about capability based security recently provided by Miller and Shapiro [MS03], we concluded that modeling collaborative behavior is crucial when modeling capabilities accurately. When propagating authority from one subject to another, the authority reducing behavior of the subjects involved should be taken into account.

We explain in Section 4 that this collaborative aspect is underdeveloped in classical Take-Grant configurations, where only two kinds of subject behavior are considered: active vs. passive. We then describe three consecutive steps to refine this formalism. We present every step in its own section: Sections 5 to 7. Every consecutive refinement will build upon the previous one: avoiding its drawbacks and adding expressive power where necessary while keeping the safety properties tractable.

As a first step, we model collaborative behavior in Section 5 by annotating every subject with a set of properties. Each property describes three orthogonal aspects of collaborative behavior:

- the possibility of initiating a collaboration (invoking behavior) vs. responding (being invoked)
- the possibility of exchanging capabilities vs. data (information)
- the possibility of providing something during the collaboration (the emitting subject or emitter) vs. accepting something (the collecting subject or collector).

These three orthogonal aspects result in eight distinctive properties (e.g. possibly initiate the emitting of data), the combination of which allowed us to model 256 different types of behavior, including both types that are available in Take-Grant systems. While the resulting formalism had gained considerable expressiveness, it soon became clear that to model many relevant problems and patterns further refinement would be required.

We tried several approaches to make behavior compositional (to build arbitrarily complex subject behavior from configurations of simple subjects), and