Abstract : A Formal Executable Semantics of Java Card

Gilles Barthe[1], Guillaume Dufay[1], Line Jakubiec[2], Bernard Serpette[1], and Simo Melo de Sousa[1,3].

[1] INRIA Sophia-Antipolis, France
[2] Université de la Méditerranée, Marseille, France
[3] Universidade da Beira Interior, Covilhã, Portugal

JavaCard is a popular programming language for multiple application smart cards. According to the JavaCard Forum this language has important features that make it the ideal choice for smart cards:

- JavaCard programs are written in a subset of Java, using the JavaCard APIs (Application Programming Interfaces). JavaCard developers can therefore benefit from the well-established Java technology;
- the JavaCard security model enables multiple applications to coexist on the same card and communicate securely, and in principle, enables new applications to be loaded on the card after its issuance.

Yet recent research has unveiled several problems in the JavaCard security model, most notably with object sharing and the associated mechanism of shareable interfaces. This has emphasised the necessity to develop environments for verifying the security of the JavaCard platform and of JavaCard programs. Thus far JavaCard security (and also Java security) has been studied mainly at two levels:

- platform level: here the goal is to prove safety properties of the language, in particular type safety and properties related to memory management;
- application level: here the goal is to prove that a specific program obeys a given property, and in particular that it satisfies a security policy, for example based on information flow.

Over the last few years, both fields have been the subject of intensive investigations. Despite impressive progress, much work remains to be done. In particular, there is no complete formalisation of the JavaCard platform as yet nor widely used tools to verify applets' properties. Besides, we do not know of any environment that supports verification both at platform and application levels.

The main contributions reported here are:

1. a formal executable specification of two crucial JavaCard platform components, namely the Java Card Virtual Machine and the ByteCode Verifier;
2. a machine-checked proof of correctness of the ByteCode Verifier. Both formalisations and proofs have been carried out in the proof assistant Coq.

The salient features of our formal specification are:

- *executability* Our formal semantics may be executed on any JavaCard program (provided it does not use native methods or APIs) and its behaviour can be confronted to reference implementations, in this case Sun's implementation of the JavaCard Virtual Machine. We view executability as a crucial asset for reliability and, in our opinion, a formal operational semantics for a (realistic) programming language must be executable;
- *completeness* Our virtual machine is complete in the sense that it treats the whole set of JavaCard instructions and it considers all the important aspects of the platform, including the firewall mechanism around which JavaCard security is organised. Our ByteCode Verifier handles the whole set of instructions but (1) it does not treat object initialisation; (2) subroutines are treated in a somewhat restrictive way.
- *suitability for reasoning* Our formalisation may be used to reason about the JavaCard platform itself, as shown here, but also to prove properties of JavaCard programs. In particular, our formalisation is well-suited to reason about security properties formulated as temporal-logic properties over execution traces.

Thus our development offers the most comprehensive to-date machine-checked account of the JavaCard platform and compares well to similar efforts carried out