Verification Methods for Software Security and Correctness

Gilles Barthe
INRIA, Sophia-Antipolis, France

The objective of the lectures is to present type-based and logic-based mechanisms to ensure reliability and security of mobile code.

In the first part of the lectures, I shall present two enforcement mechanisms for low-level code. The first mechanism is a verifier for ensuring information flow policies for confidentiality: it is a type-base mechanism, compatible with the principles of Java lightweight bytecode verification. The second mechanism is a verification condition generator for programs specified with logical annotations (pre-conditions, post-conditions, etc). It is a logic-based mechanism that permits to verify complex security policies as well as functional correctness, and is used in the context of Proof Carrying Code. Due to their complexity and to their important role in the Trusted Computing Base, it is central that these mechanisms are shown correct with the highest confidence; I shall also describe how these mechanisms have been certified using the proof assistant Coq. In the second part of the lectures, I shall relate these two enforcement mechanisms to their counterparts for source code programs. In the case of information flow typing, I shall show that non-optimizing compilers preserve typing, i.e. transform source programs typable with an information flow system into target programs that are accepted by an information flow bytecode verifier. In the case of functional correctness, I shall show that non-optimizing compilers preserve proof obligations, and discuss approaches to stretch the results to optimizing compilers.

The material presented in these lectures is partially based on the results of the EU project Mobius; its web site, see [5], contains pointers to many relevant articles and Coq formalizations.

The following articles provide good background material for the lectures.

References

1. Xavier Leroy. *Java Bytecode Verification: Algorithms and Formalizations*, Journal of Automated Reasoning, 30(3/4), pp. 235-269, 2003.
2. George Necula. *Proof Carrying Code*, Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 106-119, 1997.
3. A. Sabelfeld and A. Myers. *Language-Based Information-Flow Security*, IEEE Journal on Selected Areas in Communications, 21(1), 2003.
4. Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development: Coq'Art*, Springer Verlag, EATCS Texts in Theoretical Computer Science, 2004
5. `http://mobius.inria.fr`