Slicing Guarantees Information Flow Noninterference

Daniel Wasserrab

March 12, 2013

Abstract

In this contribution, we show how correctness proofs for intra- [7] and interprocedural slicing [8] can be used to prove that slicing is able to guarantee information flow noninterference. Moreover, we also illustrate how to lift the control flow graphs of the respective frameworks such that they fulfil the additional assumptions needed in the noninterference proofs. A detailed description of the intraprocedural proof and its interplay with the slicing framework can be found in [9].

1 Introduction

Information Flow Control (IFC) encompasses algorithms which determines if a given program leaks secret information to public entities. The major group are so called IFC type systems, where well-typed means that the respective program is secure. Several IFC type systems have been verified in proof assistants, e.g. see [1, 2, 5, 3, 6].

However, type systems have some drawbacks which can lead to false alarms. To overcome this problem, an IFC approach basing on slicing has been developed [4], which can significantly reduce the amount of false alarms. This contribution presents the first machine-checked proof that slicing is able to guarantee IFC noninterference. It bases on previously published machine-checked correctness proofs for slicing [7, 8]. Details for the intraprocedural case can be found in [9].

References

[1] G. Barthe and L. P. Nieto. Secure information flow for a concurrent language with scheduling. *Journal of Computer Security*, 15(6):647–689, 2007.

[2] G. Barthe, D. Pichardie, and T. Rezk. A certified lightweight non-interference Java bytecode verifier. In *ESOP 2007*, volume 4421 of *LNCS*, pages 125–140. Springer, 2007.

[3] L. Beringer and M. Hofmann. Secure information flow and program logics. In *Archive of Formal Proofs*. http://afp.sf.net/entries/SIFPL.shtml, November 2008. Formal proof development.

[4] C. Hammer and G. Snelting. Flow-sensitive, context-sensitive, and object-sensitive information flow control based on program dependence graphs. *International Journal of Information Security*, 8(6):399–422, 2009.

[5] F. Kammüller. Formalizing non-interference for a simple bytecode language in Coq. *Formal Aspects of Computing*, 20(3):259–275, 2008.

[6] G. Snelting and D. Wasserrab. A correctness proof for the Volpano/Smith security typing system. In G. Klein, T. Nipkow, and L. Paulson, editors, *Archive of Formal Proofs*. http://afp.sf.net/entries/VolpanoSmith.shtml, September 2008. Formal proof development.

[7] D. Wasserrab. Towards certified slicing. In G. Klein, T. Nipkow, and L. Paulson, editors, *Archive of Formal Proofs*. http://afp.sf.net/entries/Slicing.shtml, September 2008. Formal proof development.

[8] D. Wasserrab. Backing up slicing: Verifying the interprocedural two-phase Horwitz-Reps-Binkley slicer. In *Archive of Formal Proofs*. http://afp.sf.net/entries/HRB-Slicing.shtml, September 2009. Formal proof development.

[9] D. Wasserrab, D. Lohner, and G. Snelting. On PDG-based noninterference and its modular proof. In *Proc. of PLAS '09*, pages 31–44. ACM, June 2009.