Enforcing information flow policies by a three-valued analysis

Josée Desharnais
josee.desharnais@ift.ulaval.ca

Erwanne P. Kanyabwero
erwamme-pamela.kanyabwero.1@ulaval.ca

Nadia Tawbi
nadia.tawbi@ift.ulaval.ca

Université Laval, Québec, Canada

Abstract

Today, security issues are a major concern, especially when it comes to securing information flow. How can we be sure that a program using a credit card number will not leak this information to an unauthorized person? Or that one that verifies a secret password to authenticate a user will not write it in a file with public access? Those are examples of information flow breaches in a program that should be controlled. Secure information flow analysis is a technique used to prevent data misuse.

Our objective is to take advantage of the combination of static and dynamic analysis. We design a three-valued type system to statically check non-interference for a simple imperative programming language. To the usual two main security levels, public and private, we add a third value, *unknown*, that captures the possibility that we may not know, before execution, whether the information is public or private. Standard two-valued analysis has no choice but to be pessimistic with uncertainty and hence generate false positive alarms. If uncertainty arises during the analysis, we tag the instruction in cause: in a second step, instrumentation at every such point will allow us to head to a more precise result than purely static approaches. We get reduced false alarms, while introducing a light runtime overhead by instrumenting only when there is a need for it.

The goal of a security analysis is to ensure non-interference, that is, to prevent inadvertent information leaks from private channels to public channels. More precisely, in our case, the goal is 1) to ensure that, in a well-typed program, operations involving private channels do not have any effect on the content of public channels, 2) to ensure that a program not satisfying non-interference is rejected, and finally, specific to our analysis, 3) to detect a need of instrumentation in uncertain cases. Furthermore, we consider that programs have interaction with an external environment through communication *channels*, i.e., objects through which a program can get information from users (printing screen, file, network, etc.). In contrast with the work of Volpano et al. [4], variables are not necessarily channels, they are local and hence their security type is allowed to change throughout the program. This is similar to flow-sensitive typing approaches of Hunt and Sands, or Russo and Sabelfeld [2, 3]. Our approach distinguishes clearly communicating channels, through which the program interacts and which have a priori security levels, from variables, used locally. Therefore, our definition of non-interference applies to communication channels: someone observing the communication channels cannot deduce anything about the initial channels of higher security level.

We aim at protecting against two types of flows, as explained in [1]: *explicit flow* occurs when the content of a variable is directly transferred to another variable, whereas *implicit flow* happens when the content assigned to a variable depends on another variable, i.e., the guard of a conditional structure. Thus, the security requirements are first, explicit flows from a variable to a channel of lower security are forbidden and second, implicit flows where the guard contains a variable of higher security than the variables assigned, are forbidden.

References

[1] Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19:236–243, May 1976.

[2] Sebastian Hunt and David Sands. On flow-sensitive security types. In *Proceedings of the 33rd ACM Symposium on Principles of Programming Languages*, January 2006.

[3] Alejandro Russo and Andrei Sabelfeld. Dynamic vs. static flow-sensitive security analysis. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium*, pages 186–199, 2010.

[4] Dennis Volpano, Cynthia Irvine, and Geoffrey Smith. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(2-3):167–187, January 1996.