Secure compilation-with the compiler, not against: first experiments on 'Tracing LLVM'

Sébastien Michelland*[1]

[1]Laboratoire de Conception et d'Intégration des Systèmes (LCIS) – Université Grenoble Alpes, Institut Polytechnique de Grenoble - Grenoble Institute of Technology – 50, rue Barthélémy de Laffemas BP54 26902 VALENCE Cedex 09 France, France

Résumé

Countermeasures against fault injection or side-channels attacks that have software components all face the same tension. On one hand, they need fine low-level control of the program to defeat accurate, micro-architectural attack models with the help of hardware. On the other hand, security requirements are application-specific and originate in the source code (usually C). Countermeasures found in literature almost never address this abstraction gap, usually focusing on one side and dealing with the other with compiler tricks that happen to work often but fail whenever the compiler is "too smart".

I will discuss how these abstraction challenges (more than optimizations) are at the core of secure compilation, and introduce Tracing LLVM, a modified compiler for writing security countermeasures with a language approach. Tracing LLVM aims to provide preservation guarantees for small language fragments, which can then be used to write security-sensitive code, like a security toolbox added to a production compiler. I will also touch on the need for vertical integration of security from software to micro-architecture (from a software point of view) and the compiler's role in it.

*Intervenant