What is secure compilation?

Cătălin Hriţcu

Inria Paris

What is secure compilation?

- *"Secure compilation aims to preserve high-level language abstractions in compiled code, even against adversarial low-level contexts."*

  – Secure Compilation Meeting website

- **Fully abstract compilation et al fit this intuition**
- **Show 2 other properties fitting this intuition**

  – that provide practically motivated attacker models

- **Secure compilation more than full abstraction**

Side-channel attacks

- Low-level contexts can easily **observe time**

enough

- **Very powerful attacker (but realistic!)**
  - can observe the executed branches (instruction caches)
  - can observe the memory access patterns (data caches)
- **Achieving full abstraction against such a powerful low-level attacker seems hopeless**
  - high-level contexts can't observe low-level time
  - very hard to prevent low-level contexts from observing time (no concurrency, no external communication, ...)

What can we do?

- **Option 0:** deny/ignore/postpone the problem, stick with full abstraction and weak attackers

- **Option 1:** defend against side-channel attacks
- **Option 2**: devise weaker secure compilation properties that are immune to side-channels

I'll focus on these, but there might be more options

Option 1: defend against side-channels

- **Hopeless:** preserving observational equivalence of **two arbitrary programs**
- **More realistic:**
  - single program with clearly identified secrets
  - program is constant time with respect to secrets
    - no secret dependent branches or memory accesses
- **Property:** compiler preserves constant time
  - easy to achieve using **existing** compilers
- **Limited scope:** constant-time cryptography

Option 2: devise weaker property that is immune to side-channels

- **Hopeless:** preserving **observational equivalence** of two arbitrary programs
- **What's left if one gives up confidentiality?**
- **Property: robust compilation**
  - **preservation of safety in adversarial context (robust safety)**
  - **conjectures:** strictly stronger than compiler correctness
    - strictly weaker than full abstraction + compiler correctness
  - **less extensional** than FA**,** but **achievable** and **still useful**: preservation of **data invariants** and other **integrity properties**

Let's take a broad view on secure compilation

- **Different security goals / attacker models**
  - Fully abstract compilation and variants, **constant time preservation, robust compilation, ...**
- **Different enforcement mechanisms**
  - static analysis, software rewriting, reference monitors, secure hardware, randomization, ...
- **Different proof techniques**
  - logical relations, bisimulation, multi-language semantics, embedded interpreters, ...