Sustainable Compiler Verification

Tjark Weber et al.
tjark.weber@it.uu.se

UU Cybersecurity Workshop
January 12, 2024

Background: Compiler Correctness

Applications written in high-level languages are only as secure as the compiler that translates them into machine code: "Compiler-introduced security bugs are common and may have serious security impacts."[1]

Verified compilers, such as CompCert and CakeML, ensure that the generated code adheres to the high-level semantics. However, the vast majority of compilers used in production today are not verified.

[1] Jianhao Xu et al.: *Silent Bugs Matter: A Study of Compiler-Introduced Security Bugs.* USENIX Security '23.

Background: Erlang

Erlang is a programming language and runtime system designed for distributed, fault-tolerant and highly available applications. Erlang-powered nodes handle over 90% of all Internet traffic.

Erlang is an untyped language (like Python, JavaScript, . . .). This makes it difficult to reason about Erlang code at compile time, and to establish the correctness of certain compiler transformations.

Idea

To verify (only) the most critical transformations and optimizations in the Erlang compiler, thereby

- ensuring their correctness, and
- allowing more aggressive optimizations to be added with confidence.

✓ Less work than full compiler verification

✓ Can be maintained by Ericsson engineers

Plan

- A domain-specific language to express compiler transformations
- A formal model of the run-time state of Erlang programs
- Verification of transformations with the help of automated provers
- Integration into the Erlang tool-chain