Job openings

(back (index.html))

I am currently recruiting for a **fully-funded PhD studentship** and/or **postodoc studentships**, supported by by the ONR N00014-17-1-2945 Vertica project and by a Google Research Fellowship award, on the following research projects. Positions are open until fulfilled. Please, do get in touch (http://www.di.ens.fr/~zappa/index.html) if you are interested.

Validation and Synthesis of DWARF Debugging Information

> *Sorry, but last time was too f***ing painful. The whole (and *only*) point of unwinders is to make debugging easy when a bug occurs. But the f***ing dwarf unwinder had bugs itself, or our dwarf information had bugs, and in either case it actually turned several "trivial" bugs into a total undebuggable hell.*
>
> Linus Torvalds, Linux Kernel Mailing List (https://lkml.org/lkml/2012/2/10/356), Oct. 2012

DWARF is a standardised debugging data format; it is part of the ELF and Mach-O binary formats, and as such it is widely used. DWARF is obviously relied upon by debuggers, but it plays an unexpected role in the runtime of high-level programming languages and in the implementation of program analysis tools. For instance the C++ runtime relies on DWARF's `.debug_frame` table to unwind the stack and implement C++ exceptions, while program analysis tools require the `.debug_types` table to reconstruct the initial state of a program from its binary.

Generating the DWARF tables tends to be a burden for compiler authors, as each optimisation pass potentially invalidates several of them; keeping tables and code synchronised pass after pass requires a tedious and error prone logic to be added to the already twisty optimiser passes. In practice there are bugs in the generated tables: these hard to detect because DWARF informations never get any rigorous testing. Even worse, DWARF tables include scripts expressed in a Turing complete bytecode: a malicious attacker, by injecting crafted DWARF tables in a binary, can create powerful and unsuspecting trojans.

In this project we will develop theories and tools to **cross-check binaries against their DWARF tables**, and develop techniques to **synthesise correct-by-construction DWARF tables**.

The resulting tools have the potential to be routinely used to validate software and compilers and *make an impact in the real world*: Linus Torvalds would not complain about DWARF anymore!

More ambitiously, given correct tables, we can go further and investigate techniques to validate **compiler correctness** for mainstream compilers by relying on debug informations to match source and compiled code across compiler passes and optimizations.

Last update: 11/6/2018