Title: **Ensuring Control-Flow Integrity in presence of Faults Injection**

Supervisors: **Cristian Ene, Marie-Laure Potet (Vérimag)**
(e-mail: Prenom.Non@univ-grenoble-alpes.fr)

Fault attacks consist in changing the device's behavior by using a laser beam or applying voltage, clock or electromagnetic glitches. Their goal is to change the correct progress of the algorithm and hence, either to allow gaining some privilege access or to allow retrieving some secret information based on an analysis of the deviation of the corrupted behavior with respect to the original one.

Countermeasures have been proposed to protect embedded systems against faults, both in hardware and in software. Fault detection is generally based on spatial, temporal or information redundancy at hardware or software level. At the software level, code hardening can be performed at source level, in dedicated compilation passes on some intermediary assembly code, or at link-time, using a binary rewriting tool.

A basic property targeted by most of these countermeasures is Control-Flow Integrity[1]: the goal of the adversary is to hijack the control flow of the program. In the literature, there is no clear formal definition of Control-Flow Integrity, but rather a hierarchy of Control-Flow Integrity related properties. The goal can be:

- to enforce the program execution to follow only paths existing in the Control Flow Graph (CFG)[2]. The added overhead is rather low in this case, but there is no guarantee about the execution of each single instruction.
- to enable the detection of any attack that disrupts the control flow by not executing at least two adjacent statements of the code at the granularity of single C statements [3]. But the price to pay is a significant performance overhead.
- to ensure that loops always perform the expected number of iterations[4]; this is a compromise between the two above properties and allows to limit the performance overhead.

The purpose of this internship is ;

1. first, to formalize the various Control-Flow Integrity related properties;
2. to compare the existing proposed countermeasures for Control-Flow Integrity;
3. to propose a new countermeasure (or to adapt an existing one) and to implement it at the level of LLVM's Intermediate Representation as a compilation pass.
4. to validate the implemented countermeasure either formally or experimentally using the Lazart tool, which is a tool developed at Verimag, allowing to simulate fault injections and to check the program robustness.

Biblio

[1] Control-Flow Integrity - M. Abadi, M. Budiu, U. Erlingsson and J. Ligatti - CCS 2015.

[2] Picon: Control flow integrity on llvm ir - Coudray T, Fontaine A, Chifflier P.
In Symposium sur la securite des technologies de l'information et des communications 2015.

[3] Formally verified software countermeasures for control-flow integrity of smart card C code - Heydemann, Karine, Jean-François Lalande, and Pascal Berthomé. Computers & Security 85 (2019): 202-224.

[4] Compiler-assisted loop hardening against fault attacks -
Proy, Julien, Karine Heydemann, Alexandre Berzati, and Albert Cohen.

ACM Transactions on Architecture and Code Optimization (TACO) 14, no. 4 (2017): 36.

[5] Lazart: A Symbolic Approach for Evaluation the Robustness of Secured Codes against Control Flow Injections
M.L Potet, L. Mounier, M. Puys and L. Dureuil, ICST 2014