Internship: Static analysis of programs for micro-architecture aware fault models

Reference	CYBERINSTITUTE-INT-20006
Description	

Research theme

Embedded systems are made of both hardware and software components. They are designed to perform some tasks while ensuring security properties. For example, a smart card must check a PIN code before performing a sensitive operation. It is of utter importance that it's absolutely impossible to perform the sensitive operation if the PIN code hasn't been verified.

Even if the system is known to be correct (it does not contain bugs) one can try to disrupt its behavior by physically acting on it: this is called a **fault injection attack**. A "popular" technique is to use a laser to light a precise point of the circuit at a precise time, hoping to change the value of a program variable in memory or of one control-flow condition.

Designers of such systems have to include **countermeasures** (which can be hardware or software based) to protect them from this kind of attacks. However, they increase the cost of the system and it's not easy to assert their adequacy. They may not thwart all attacks, or some may not be necessary (faults that do not result in a security violation should be ignored).

Technical context

In order to assess the robustness of a system against fault injection attacks, **software fault models** are often used. They provide a software level description of the effects the hardware injection can produce. Such fault models allow to analyze at the software level the effects of attacks on the system. A classic approach is to *instrument* the program to get a "mutant": a modified program whose behavior is the one of the program when the fault is injected.

This mutant can either be compiled and executed (to check the effect of a fault on a specific execution) or analyzed with a static analysis tool [TL-face,Lazart]. Static analysis is a set of techniques allowing to check some properties of a program. Contrary to testing, in which the program is run repeatedly on a limited set of various inputs, static analysis allows proving properties holding for all possible executions.

In his PhD. thesis work, Johan Laurent [thèse] has shown that *classic*, generic software fault models, such as "modify the value of a variable", or "test inversion" don't include all possible faulted behaviors [modèles]. These generic models ignore the processor micro-archicture (i.e. the "details" of the processor mechanisms, such as the *pipeline* or *speculative execution*, hidden registers) that can create more complex faulted behaviors and so new fault models.

In order to use these micro-architecture aware fault models, one must describe the assembly program execution, including some aspects of the processor micro-architecture.

Objectives

We have developed a tool that takes a binary application program and produces a mutant describing the steps of the (faulted) program execution on a RISC-V micro-architecture [outil].

This mutant is a C program. It can be directly compiled and executed for testing a fault on a specific execution, or analyzed with a static analysis tool. Since it's not practically possible to execute the program on all possible inputs, a complete assessment necessarily requires the use of static analysis.

Static analysis is even more important if one considers the effect of *multiple faults* (the attacker can inject several faults in order to e.g. disable counter-measures). The number of cases to consider mandates analysis at a more global scale.

Due to fundamental indecidability problems (Rice's theorem) static analysis techniques are necessarily incomplete. Such analysis can not always give a result for a given property of a given program. In particular, the various static analysis techniques are often very sensitive on the form of the analyzed program or the type of properties considered. The current mutant generated by our tool is well suited to the *value analysis* technique of the frama-C tool [frama-c]. However, we want to make possible its use on other static analysis techniques. This project is a collaboration with the PACSS team of Verimag laboratory, that has developed the concolic analysis-based tool Lazart [lazart].

The work will consist in:

1. Identify the limits of the mutant when used in other static analysis tools (such as the concolic analysis tool Lazart).

   Propose structural modifications in order to enhance the results provided by these types of analysis.

2. Study how multiple faults can be dealt with using Lazart (or other tools): same faults at different times, different faults at

different times.

Context

This work is part of the CODEC [codec] project (HW/SW CO-DEsign of processor Countermeasures) involving the CTSYS team of LCIS laboratory and the PACSS team of Verimag laboratory. The physical location is LCIS in Valence, with a strong interaction with Verimag in Grenoble.

The Grenoble Alpes Cybersecurity Institute – in short, Cyber@Alps – is a project selected in 2017 by the Cross-Disciplinary Program (CDP) of the IDEX Univ. Grenoble Alpes and aims at undertaking ground-breaking interdisciplinary research in order to address cybersecurity and privacy protection challenges. Our main technical focus are on cost effective secure elements, security of critical infrastructures all along their life cycle, vulnerability analysis and global challenges in terms of risk analysis and validation of large systems, including practical resilience across the industry and the society. Our approach to cybersecurity is holistic, encompassing technical, legal, law-enforcement, economic, social, diplomatic, military and intelligence-related aspects with strong partnerships with the private sector and robust national and international cooperation with leading institutions in France and abroad (https://cybersecurity.univ-grenoble-alpes.fr)

References

- Model: Johan Laurent, Vincent Beroulle, Christophe Deleuze, Florian Pebay-Peyroula, Athanasios Papadimitriou. On the Importance of Analysing Microarchitecture for Accurate Software Fault Models, 2018 21st Euromicro Conference on Digital System Design (DSD), Aug 2018, Prague, France. hal-01899800
- PhD: Evaluation par prototypage virtuel et analyse statique de la sécurité des systèmes matériels et logiciels face aux attaques laser. Thèse de doctorat, Johan Laurent, en cours.
- Tool: Johan Laurent, Christophe Deleuze, Vincent Beroulle, Florian Pebay-Peyroula. Analyzing Software Security Against Complex Fault Models with Frama-C Value Analysis, 2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), August 2019
- frama-C: <<framac>> logiciel libre d'analyse statique http://frama-c.com
- TL-face: <<tlface>> Formal verification of a CRT-RSA implementation against fault attacks, M. Christofi, B. Chetali, L. Goubin, D. Vigilant, Journal of Cryptographic Engineering, Vol 3, Issue 3, pp 157-167, 2013
- Lazart: <<lazart>> Lazart: A Symbolic Approach for Evaluation the Robustness of Secured Codes against Control Flow Injections, M . Potet, L. Mounier, M. Puys, L. Dureuil, IEEE Seventh International Conference on Software Testing, Verification and Validation (ICST), 2014
- PACSS: http://www-verimag.imag.fr/PACSS.html
- CODEC: HW/SW CO-DEsign of processor Countermeasures, https://cybersecurity.univ-grenoble-alpes.fr/research-topics/software-vulnerabilities/codec-selected-by-irs-call-for-projects-univ-grenoble-alpes--794742.htm

Prerequisites	Master in computer science or embedded systems. Some previous exposure to static analysis tools, or knowledge about computer architecture (pipeline, speculative execution) or with the Haskell language will be appreciated.

Tutors	Christophe Deleuze
Applications	Please send your resume, application letter with two recommendations (including education director), first year master's degree grades (mandatory) and second year grades (if possible) to cyberalps-contact@univ-grenoble-alpes.fr For more information on the internship, please contact christophe.deleuze@lcis.grenoble-inp.fr
Location	Valence
Starting date	02/2020
Duration	5 or 6 months
Allowance	In accordance with existing regulations (approx. 560€/month). Part of travel expenses can be covered.