Evaluation with Virtual Prototyping and Static Analysis
of the Security of Hardware and Software Systems against Laser Attacks

Key words: fault attacks, security certification, fault simulation, static analysis, virtual prototyping

Laser allows attackers to realize very efficient attacks which threaten the security of digital systems [FA06]. These attacks consist in using a laser to inject errors at specific times targeting specific components in the digital system. This digital system is a hardware architecture which executes a software application. The injected errors can then be propagated or masked, both by the hardware and the software parts of the system. The errors which are propagated can generate security breaches. Security certifications experimentally verify the vulnerabilities of the hardware part and of the application the system executes against different types of attacks (including the laser ones). To pass these certifications it is necessary to anticipate from the early design step the design of countermeasures (hardware and/or software) against these attacks. In fact, late modifications increase the system costs and its time to market.

The thesis goal is to propose a new approach to evaluate and enhance the security of embedded systems against laser attacks, considering very early in the development flow conjointly the hardware architecture and its application.

To achieve this work early in the development flow, it is necessary to consider virtual prototypes rather than hardware prototypes. However, the use of virtual prototypes to inject faults implies high simulation times. Thus, the analyses of the fault propagations or fault masking in the hardware and software parts will be done using static analysis methods.

The work will be divided in two main steps: (1) the inference of a high level fault model for a RTL description of a hardware architecture and for a given application and (2) the use of this fault model coupled with security assertions to determine by static analysis the potential breaches of the system. Once the errors potentially generating breaches will be known (and a high number of faults filtered by the static analyses), it will be possible to accurately simulate these faults. This RTL simulation will allow us to observe the error effects and then to propose efficient and low cost countermeasures.

The first step consists in creating a fault model using several abstraction levels (i.e. “cross layer fault model”). Indeed, the software fault models (e.g. instruction jump, data flow modifications, etc.) which do not rely on any hardware information, are generally not realistic. These unrealistic models drive designers to under or over-estimate their countermeasures. But fault simulation at the RTL level, which injects more representative errors, is too time consuming to allow a complete application simulation. Thus, the objective of this first step is to give to the system developer a high level fault model taking into account both the characteristics of the laser, the hardware architecture, and the executed application.

The classical RTL transient fault models (single or multi bit-flips, etc.) do not take into account the laser locality characteristic. So, the step 1 will reuse the RTL fault model proposed by A. Papadimitriou [PA16] which takes into account the laser locality characteristic. Using this model, we will determine (by RTL

simulation) the effects of the faults on the processor architecture for a given executable. These effects simulated at the RTL level for each instruction or sequence of instructions will be modelled by software faults at the level of the source code (e.g. mutations and saboteurs).

The second step aims at finding out what security breaches the faults can create. Firstly, it is necessary to express the security needs of the application. These can be expressed as security assertions inserted in the application source code [ACSL]. Such an assertion formalize, at a given program point, a property of the program state ensuring the absence of security breaches.

Secondly, one has to identify, among all the faults from the model, which faults may violate at least one security assertion. Due to combinatorial explosion, it is not realistic to "test" the program for each possible parameter value and each possible fault (or combination of faults). Based on careful approximations, static analysis techniques allow reasoning on sets of executions including various parameter values and fault occurrences. Such techniques have been studied for a long time. Their usage is growing to prove program properties such as the absence of certain kinds of bugs. Mature tools such as [frama-C] have been developed. More recently, they have started to be used in the context of the analysis of fault attack effects on a system or program [CH13,PO14].

Student diploma: Master in embedded system, or in computer science, or in microelectronic

Skills: compilation, computer architectures, prototyping and simulation

Location : Grenoble INP LCIS, Valence

Contacts : Christophe.deleuze@lcis.grenoble-inp.fr; vincent.beroulle@lcis.grenoble-inp.fr

References

[ACSL] ANSI/ISO C Specification Language, http://frama-c.com/acsl.html

[CH13] 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

[frama-C] Open source static analysis tool, http://frama-c.com/

[FA06] The sorcerer's apprentice guide to fault attacks, H. Bar-El ; H. Choukri ; D. Naccache ; M. Tunstall ; C. Whelan, proceding of the IEEE, Vol. 94, Issue 2, pp. 370-382, 2006

[PA16] Thèse de Athanasios Papadimitriou, « modélisation au niveau RTL des attaques laser pour l'évaluation des circuits intégrés sécurisés et la conception de contremesures », sous la direction de Vincent Beroulle, David Hély, et Paolo Maistri, Laboratoire Grenoble INP-LCIS, Univ. Grenobles Alpes, 2016.

[PO14] 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