

JAIF 2019

Journée thématique sur les attaques par injection de fautes Bilan de la journée et perspectives

Guillaume Bouffard, Damien Couroussé, Sylvain Guillet, Karine Heydemann, Marie-Laure Potet

August 23, 2019

Contents

1	Introduction	1
1.1	Objectifs de la journée	1
1.2	Comité d'organisation	2
2	Soutien financier et institutionnel	2
3	Participants	2
4	Retours sur le sondage post-journée	3
5	Programme	3
6	Résumés des présentations	4
6.1	Injection de fautes par médium EM : modèle et implications	4
6.2	On-the-fly laser-induced corruption of the firmware stored into the flash memory of a 32-bit microcontroller	5
6.3	How modern System-on-Chips are vulnerable to fault attacks	5
6.4	Analyse de fautes au niveau RTL	5
6.5	IntrinSec: an intrinsically secure RISC V processor	6
6.6	Certification et IoT	6
6.7	Concevoir des applications robustes à l'injection de fautes (projet CLAPs)	6
6.8	Compilation de contre-mesures	6
6.9	Sécurisation automatisée des boucles à la compilation	6
6.10	Techniques d'analyse statique pour détecter des vulnérabilités sécuritaires lors d'une revue de code	7
6.11	Évaluation sécuritaire de code binaire soumis à des attaques en faute	7

1 Introduction

Ce document fait le bilan de la journée thématique sur les attaques par injection de fautes qui s'est tenue le 23 mai 2019 à Minatec, Grenoble.

Le site internet mise en place pour la journée se trouve ici : <https://jaif2019.github.io>

1.1 Objectifs de la journée

Cette journée s'inscrivait dans la suite de la journée SERTIF organisée en 2016 à Grenoble, puis de la journée JAIF organisée en 2018 à Paris.

Le workshop avait pour objectif de réunir la communauté de la recherche française en analyse de fautes sur des systèmes de sécurité, pour consolider ce savoir-faire et pour faire émerger des recherches plus globales. Notre communauté est très avancée sur le plan mondial, et travaille sur des aspects variés :

- attaques à l'aide d'injection de fautes, et modèles de fautes,
- simulation et méthodes formelles pour évaluer la robustesse d'un système à une attaque par injection de fautes, et pour comprendre l'impact des fautes,
- preuve de sécurité,
- conception de contremesures matérielles ou logicielles,
- application outillée de contremesures,
- etc.

Le contexte des systèmes embarqués sécurisés évolue rapidement, et il est important d'envisager l'évolution des attaques mais aussi des systèmes. On assiste aujourd'hui à un changement de paradigme pour aller de systèmes fermés, tels que la carte à puce, vers des systèmes ouverts qui embarquent des enclaves sécurisées (e.g. Trusted Execution Environments). Également, les attaques exploitant les injections de fautes deviennent plus complexes, avec par exemple la possibilité de réaliser des injections multiples ou la combinaison avec des observations de canaux cachés.

Le programme de cette journée est construit de sorte à favoriser les échanges entre les participants sur l'ensemble de ces sujets, au travers de présentations et de discussions.

1.2 Comité d'organisation

Le comité d'organisation se composait des personnes suivantes:

- Guillaume Bouffard, ANSSI
- Damien Couroussé, CEA (chair)
- Sylvain Guilley, Telecom ParisTech / Secure-IC
- Karine Heydemann, Sorbonne Université / LIP6
- Marie-Laure Potet, VÉRIMAG, Grenoble Alpes Cybersecurity Institute

2 Soutien financier et institutionnel

Cette journée était organisée avec le soutien financier de :

- l'IRT NanoElec, dans le cadre du programme Pulse
- Cybersecurity Institute Grenoble
- l'ANSSI

Cette journée thématique était associée au GT Sécurité des Systèmes Matériels, qui est commun au GdR SOC2 et au GdR Sécurité Informatique.

La Table 1 détaille les subventions reçues pour la journée, et le principaux postes de dépenses.

3 Participants

Nous avons eu 128 inscriptions à la journée,

- dont 53 personnes basées en région grenobloise.
- 35 participants étaient industriels,

Table 1: Synthèse du budget de la journée

	CEA	VERIMAG	notes
Subvention			
Sponsoring IRT NanoElec	4 000 €		
Sponsoring ANSSI	1 000 €		
Sponsoring Cyber@Alps		1 000 €	
Total subvention	5 000 €	1 000 €	
Dépenses			
Location de la salle	1 540 €		HT. Accueil Minatec – Capacité 180 pers.
Restauration – repas et pauses	4 265 €		TTC. Devis pour 119 pers.
A/R Paris-Grenoble pour un orateur	250 €		Thomas Trouchkine
Total des dépenses	5 805 €	250 €	

- 33 participants étaient affiliés au CEA Grenoble.

Pendant la journée, nous avons compté 115 participants environ la matinée, et 100 environ l’après-midi.

4 Retours sur le sondage post-journée

TODO

5 Programme

Le programme de la journée était aménagé pour maximiser les interactions entre les participants. Un temps de questions et de discussions, commun à toutes les présentations de la session, était organisé à la fin de chaque session.

Quelques photos prises pendant la journée sont diffusées sur le site internet du workshop.

- 09:30–10:00 Accueil des participants autour d’un café
- 10:00–10:10 Introduction à la journée
- 10:10–11:25 **Session #1. Injection de fautes**
 - Philippe Maurine (LIRMM). *Injection de fautes par médium EM : modèle et implications.*
 - Brice Colombier (Univ. Saint-Étienne). *On-the-fly laser-induced corruption of the firmware stored into the flash memory of a 32-bit microcontroller.*
 - Ronan Lashermes, Thomas Trouchkine (INRIA, ANSSI). *How modern System-on-Chips are vulnerable to fault attacks.*
- 11:25–11:40 Pause
- 11:40–12:30 **Session #2. Architectures matérielles robustes**
 - Vincent Beroulle (LCIS Valence). *Analyse de fautes au niveau RTL.*
 - Olivier Savry (CEA). *IntrinSec: an intrinsically secure RISC V processor.*
 - Discussion
- 12:30–13:45 Déjeuner
- 13:45–14:35 **Session #3. Questions ouvertes sur la sécurité des systèmes**
 - Guillaume Bouffard (ANSSI). *Certification et IoT.*



Figure 1: L’assemblée des participants.

- Laurent Mounier et Marie-Laure Potet (VERIMAG). *Concevoir des applications robustes à l’injection de fautes (projet CLAPs).*
- Discussion
- 14:35–14:50 Pause
- 14:50–15:40 **Session #4. Protections logicielles**
 - François de Ferrière (STMicroelectronics). *Compilation de contre-mesures.*
 - Julien Proy (INVIA). *Sécurisation automatisée des boucles à la compilation.*
 - Discussion
- 15:40–15:55 Pause
- 15:55–16:45 **Session #5. Analyse de code**
 - David Féliot (CEA Grenoble). *Techniques d’analyse statique pour détecter des vulnérabilités sécuritaires lors d’une revue de code.*
 - Jean-Baptiste Bréjon (LIP6). *Évaluation sécuritaire de code binaire soumis à des attaques en faute.*
 - Discussion
- 16:45–16:50 Conclusion de la journée

6 Résumés des présentations

6.1 Injection de fautes par médium EM : modèle et implications

Philippe Maurine (LIRMM)

La première publication traitant d’attaques par faute(s) conduites par médium électromagnétique a été publiée en 2002. Plus de 15 ans après, le mécanisme par lequel ces fautes apparaissent n’est toujours pas clairement établi. Dans ce contexte, cette présentation s’attachera à expliquer finement l’apparition des fautes et ce en partant des principes de l’induction électromagnétique jusqu’au tréfonds des circuits intégrés. Enfin, les enseignements de ce modèle seront tirés tant pour établir des pistes de contremesures que des moyens d’améliorations des plateformes d’injection EM.

6.2 On-the-fly laser-induced corruption of the firmware stored into the flash memory of a 32-bit microcontroller

Brice Colombier (CEA), Alexandre Menu (EMSE), Jean-Max Dutertre (EMSE), Pierre-Alain Moëllic (CEA), Jean-Baptiste Rigaud (EMSE), Jean-Luc Danger (Telecom ParisTech)

L'injection de faute laser est souvent considérée comme la technique d'injection de faute la plus efficace. En effet, elle offre la plus grande précision spatiale, permettant ainsi à l'attaquant d'induire des fautes au niveau bit. Néanmoins, l'expérience acquise lors de l'attaque de cibles 8 bits n'est pas directement transférable à des microcontrôleurs 32 bits complexes, et ces attaques deviennent de plus en plus difficiles. Dans cette présentation, nous montrons que la mémoire Flash est une zone sensible à l'injection de fautes même sur des microcontrôleurs aux architectures avancées. Ces fautes ont lieu pendant la phase de lecture, et la donnée stockée n'est donc pas modifiée. Après une caractérisation des fautes réalisées et du modèle de faute associé, nous donnerons des exemples détaillés de corruption d'instructions au niveau bit et d'attaques sur des codes d'évaluation classiques. Nous proposerons finalement une hypothèse à propos des caractéristiques physiques de la micro-architecture qui permet d'expliquer le modèle de faute observé.

6.3 How modern System-on-Chips are vulnerable to fault attacks

Guillaume Bouffard (ANSSI), Sébanjila Kevin Bukasa (INRIA), Mathieu Escuteloup (INRIA), Ronan Lashermes (INRIA), Thomas Trouchkine (ANSSI)

Electromagnetic fault injection (EMFI) is a well known technique to disturb the behavior of a chip and weaken its security. Yet these attacks are mostly done on simple microcontrollers since the fault effect is relatively simple and understood.

Unlocking EMFI on modern System-on-Chips (SoCs), the fast and complex chips ubiquitous today, requires to understand the impact of the faults. In this paper we target the BCM2837 SoC, with four Cortex-A53 cores from ARM. We propose an experimental setup and a forensic process to create exploitable faults and assess their impact on the micro-architecture.

The observed behaviors are radically different to what was previously obtained on microcontrollers. Subsystems (L1 caches, L2 cache, MMU) can be individually targeted leading to new fault models. We highlight the differences in the fault impact with or without an Operation System, therefore showing the importance of the software layers in the exploitation of a fault.

The complexity and speed of a SoC does not protect them against hardware attackers, quite the contrary.

We advocate for the design of secure generic cores with a stronger security model to run all security related code (which encompass all privileged code).

6.4 Analyse de fautes au niveau RTL

Vincent Berouille (LCIS Valence)

Dans cet exposé, nous présenterons une méthode d'évaluation et d'amélioration des contremesures matérielles et logicielles pour protéger l'exécution d'un code sécurisé contre les attaques en fautes.

Afin de se protéger contre les attaques en fautes, les développeurs utilisent souvent des contremesures logicielles. Mais ces contremesures ne protègent le code que contre les effets induits par les modèles de fautes logiciels (saut d'instruction, l'inversion de test...). Or, ces modèles de fautes ne prennent pas en compte l'implémentation matérielle des processeurs. En analysant la microarchitecture au niveau RTL des processeurs, il est possible de trouver des fautes matérielles qui créent des failles de sécurité. Nous donnerons des exemples de ce type de fautes en nous appuyant sur des codes sécurisés issus de FISSC et en utilisant la description RTL d'un processeur RISC-V. Nous montrerons notamment l'importance des registres cachés dans le pipeline du processeur. Finalement, nous proposerons des contremesures logicielles robustes contre ces attaques en faute.

6.5 IntrinSec: an intrinsically secure RISC V processor

Olivier Savry (CEA)

Dans le cadre du projet Nanotrust soutenu par l'IRT Nanoelec nous développons une gamme de processeurs intrinsèquement sécurisés pour les CPS. Ces processeurs sont capables d'exécuter du code chiffré où chaque instruction est également associée à un MAC qui permet une vérification de son intégrité au runtime. Cette structure permet également la mise en place aisée d'un CFI intrinsèque avec un chaînage cryptographique des Basic Blocks et de protection contre les stack overflows. Toute déviation du graphe de flot de contrôle est ainsi détecter par une erreur à la vérification des MAC.

6.6 Certification et IoT

Guillaume Bouffard (ANSSI)

Résumé à venir.

6.7 Concevoir des applications robustes à l'injection de fautes (projet CLAPs)

Laurent Mounier et Marie-Laure Potet (VERIMAG)

Concevoir des applications robustes à l'injection de fautes est un processus complexe qui nécessite de prendre en compte les scénarios d'attaques (que veut-on protéger), l'effet des attaques (le modèle de fautes) et ceci afin de mettre en place les contre-mesures adéquates. Ce processus est rendu encore plus complexe dans le cadre du multi-fautes, qui permet en plus de modifier le comportement des contre-mesures.

Le projet CLAPs s'intéresse d'une part à proposer des analyses du code source, au code binaire jusqu'aux attaques physiques, afin de pouvoir rendre robuste une implémentation et d'autre part à proposer des contre-mesures automatiques permettant de se prémunir contre des modèles de fautes déterminés.

Nous illustrerons ces démarches sur les études de cas du projet CLAPs issues du benchmark FISSC et sur une application interne au projet, un Firmware Update.

6.8 Compilation de contre-mesures

François de Ferrière (STMicroelectronics Grenoble)

STMicroelectronics développe des outils de compilation basés sur la technologie LLVM pour ses coeurs propriétaires ainsi que pour le processeur ARM.

Afin d'ajouter des contre-mesures logicielles de résistance aux attaques par injection de fautes, qui puissent être à la fois non triviales, fiables et rapides à implémenter dans les produits développés par STMicroelectronics, nous avons implanté des techniques de génération de code pour la cybersécurité dans notre compilateur LLVM de production.

Nous présentons dans cet exposé ces techniques et transformations que nous avons implémentées. Nous montrons comment elles contribuent au renforcement de la protection des applications. Nous détaillons également comment ces techniques peuvent être appliquées localement à certaines régions critiques d'une application afin de satisfaire les contraintes industrielles de taille et de performances de ces applications.

6.9 Sécurisation automatisée des boucles à la compilation

Julien Proy (INRIA), Karine Heydemann (Univ. Sorbonne, Paris), Alexandre Berzati (INRIA), Albert Cohen (Google)

La sécurisation des systèmes embarqués est un enjeu majeur dans l'industrie. Le déploiement de contre-mesures logicielles est encore largement réalisé de façon manuelle, induisant des coûts et temps

de développement importants. Afin de réduire ces coûts, les industriels sont à la recherche d'approches automatisées, nécessitant des schémas de protection génériques.

Nous présentons dans cet exposé une contre-mesure dédiée à la sécurisation des boucles applicable automatiquement à la compilation. Une implémentation dans le compilateur LLVM ainsi qu'une étude des interactions avec les optimisations du compilateur sont également détaillées. Enfin, nous montrons les résultats associés provenant de simulations et de campagnes d'attaques physiques.

6.10 Techniques d'analyse statique pour détecter des vulnérabilités sécuritaires lors d'une revue de code

David Féliot (CEA Grenoble)

L'évaluation de la résistance aux attaques d'un produit de type carte à puce comprend une revue de code du logiciel embarqué. L'objectif de cette revue est de détecter dans le code source des vulnérabilités qui peuvent être exploitées par un attaquant pour forcer ou contourner des fonctions de sécurité, par exemple une fonction de contrôle d'accès. L'exposé présentera d'une part les spécificités et les contraintes liées à l'activité d'évaluation sécuritaire, et d'autre part l'apport des techniques d'analyse statique pour augmenter la fiabilité et l'efficacité de la revue de code.

6.11 Évaluation sécuritaire de code binaire soumis à des attaques en faute

Jean-Baptiste Bréjon (LIP6), Karine Heydemann (Univ. Sorbonne, Paris), Emmanuelle Encrenaz (Univ. Sorbonne, Paris), Quentin Meunier (Univ. Sorbonne, Paris)

Les attaques en fautes constituent une menace sérieuse pour les applications embarquées. Pour s'en prémunir, le code peut être renforcé par l'insertion de protections visant à détecter ou tolérer des attaques en faute et la robustesse obtenue doit être évaluée. Dans cet exposé, nous présenterons une approche, implantée dans le framework RobustB, combinant des analyses statiques et dynamiques de code avec de la vérification formelle et un ensemble de métriques pour évaluer la robustesse d'un code binaire soumis à des attaques en faute. Notre approche modélise la recherche de vulnérabilités par des problèmes d'équivalence-checking résolus par un SMT solver.

RobustB permet d'analyser la robustesse de code après compilation, et à l'aide des métriques, il permet de comparer des codes intégrant différentes protections et/ou compilés avec différents compilateurs et/ou différents niveaux d'optimisation. En particulier, nous illustrerons l'apport de notre approche et de ses métriques à l'analyse de vulnérabilités, l'analyse des effets des optimisations de code de compilateurs ainsi qu'à la comparaison de différentes protections combinées ou non sur des codes protégés au niveau du code source.