Sécurisation des applications contre les attaques en fautes : retour sur quelques challenges [a]

Marie-Laure Potet

Vérimag, University Grenoble Alpes, France

Journée Thématique sur les attaques par injection de fautes, 1 octobre 2025

a. supported by Grenoble Alpes IDEX project CyberAlps, Grenoble Alpes labex PERSYVAL-Lab project CLAM, PEPR Arsène, PEPR SecurEval

Outline

Journées thématiques sur les attaques par injection de fautes

2016 : Workshop Projet ASTRID SERTIF / Grenoble (J. Cledière, M-L Potet, T-H. Le)

2018 : Jussieu (K. Heydemann)
2019 : Grenoble (D. Courrousé)
2020 : ENS, Grenoble, distanciel
2021 : ENS/Paris (G. Bouffard))
2022 : Valence (V. Beroulle)
2023 : Gardennes (J(M. Dutertre)
2024 : Rennes (R. Lashermes)
2025 : Grenoble (D. Couroussé)
. . .

2016 sur invitation (exposés, participants), 2018 (ouvert), 2020 appel à sponsoring, 2021 appel à soumission

Journées thématiques sur les attaques par injection de fautes

2016 : Workshop Projet ASTRID SERTIF / Grenoble (J. Cledière, M-L Potet, T-H. Le)

2018 : Jussieu (K. Heydemann)
2019 : Grenoble (D. Courrousé)
2020 : ENS, Grenoble, distanciel
2021 : ENS/Paris (G. Bouffard))
2022 : Valence (V. Beroulle)
2023 : Gardennes (J(M. Dutertre)
2024 : Rennes (R. Lashermes)
2025 : Grenoble (D. Couroussé)
. . .

- techniques d'attaques physiques
- processus d'évaluation et outils
- design de composants securisés (HW et SW)
- nouvelles applications et chaine de confiance

2016 sur invitation (exposés, participants), 2018 (ouvert), 2020 appel à sponsoring, 2021 appel à soumission

SERTIF : les challenges

⇒ Simulation pour l'Evaluation de la RobusTesse des applications embarquées contre l'Injection de Fautes.

	Identification	Exploitation
< one hour	0	0
< one day	1	3
< one week	2	4
< one month	3	6
> one month	5	8
Not practical (see below)	*	*

Table 1: Rating for Elapsed Time

Range of values*	TOE resistant to attackers with attack potential of:
0-15	No rating
16-20	Basic
21-24	Enhanced-Basic
25-30	Moderate
31 and above	High

Table 13: Rating of vulnerabilites and TOE resistance

Challenges méthodologiques : améliorer/automatiser les processus d'évaluation, combiner analyse de code en boite blanche et attaques physiques en boite noire en prenant en compte le multi-faute

Challenges scientifiques : formalisation générique de modèle de faute prenant en compte les caractéristiques du composant, savoir formaliser le lien entre contre-mesures et attaques ; entre contre-mesures et biens à protéger

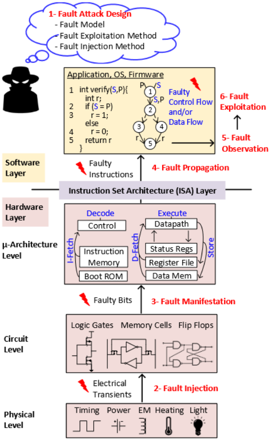
FISSC : a fault injection secure collection [SAFECOMP 2016]

Outline

Top-down or Bottom-up ?

⇒ We have to consider complementarity between source level, compilation process, binary level and physical attacks

- At the source level we track weaknesses relatively to application attack scenarios /at the binary level we track weaknesses relatively to attack technics
  - not necessary the same fault models/countermeasures
- Auditor must understand the code and identify potential exploitable paths/developper must harden this code w.r.t. assets to be protected
  - code and particularly countermeasures must be understood from source to binary levels including the compilation process

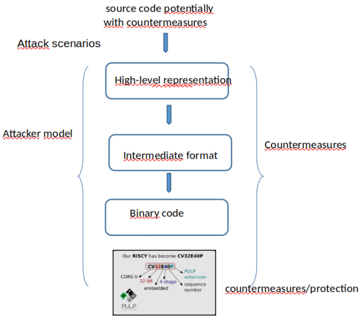


borrowed from Bilgiday Yuce

Top-down or Bottom-up ?

⇒ We have to consider complementarity between source level, compilation process, binary level and physical attacks

- At the source level we track weaknesses relatively to application attack scenarios /at the binary level we track weaknesses relatively to attack technics
  - not necessary the same fault models/countermeasures
- Auditor must understand the code and identify potential exploitable paths/developper must harden this code w.r.t. assets to be protected
  - code and particularly countermeasures must be understood from source to binary levels including the compilation process



Multi-Faults

Tools, processes and counter-measures are presently dedicated to single fault with classical fault models.

- There exists metrics for robustness evaluation
- Build robust applications is a try-and-retry process
- Countermeasures can be added in a systematic way

Multi-faults (spatial or temporal) and multi (or complex) models are now the state -of-the-art in terms of attacks.

- Evaluation becomes a very combinatorial process
- Comparing or evaluating robustness is a new problem
- Countermeasures can be also attacked and must be added judiciously

⇒ Build/Analyze robust applications becomes a very challenging problem

Outline

Countermeasures analyses

Open challenges

- Choose or build the most appropriate countermeasures
  - security/performance trade-offs
- Ensure that countermeasures are preserved by compilers
  - Combining countermeasures and fine-grained optimizations

$\Rightarrow$ how to help developers as well as auditors : determining generic countermeasures properties.

Assisting tools

- detect redundant countermeasure application [FDTC 2020]
- Classify countermeasure properties w.r.t. fault models (robustness level) [FDTC 2023], [CPP 2025]
- Adapted placements w.r.t. hot spots identification [FDTC 2023]

Countermeasure analyses methodology

Context :

- hardening consists in replacing a sensitive element (*SS*) by a protected element (*PS*)
- detecting countermeasures : a dangerous attack triggers a blocking behavior (duplication test and loading, adding counter, shadow stack . . .)

Expected properties :

- *Correctness* : *SS* can be safely replaced by *PS* (refinement)
- *Robustness :* *PS* protects against the fault model for which *SS* is sensitive (*PS* behaves as *SS* or stops the execution)

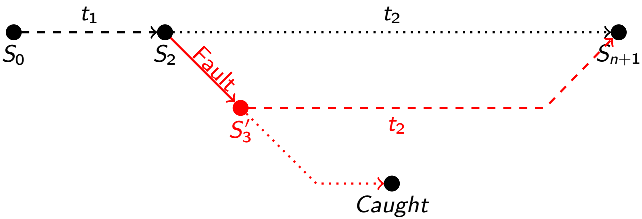
Extension for multi-fault :

- *Robustness level* :*PS* protects against a set of fault models up to the order n (*PS* behaves as *SS* or stops the execution)

$\Rightarrow$ can be established by proof (CompCert, S-monad) (correctness and 1-robustness) or by symbolic execution or combinatory exploration (Lazart, Celtic, . . .).

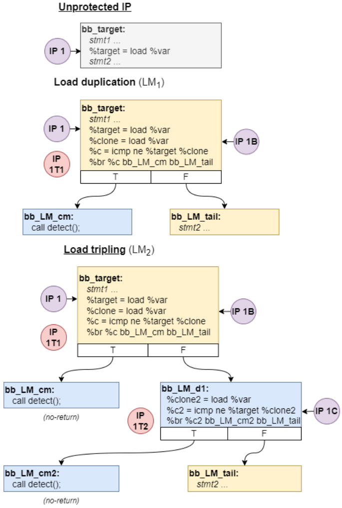
Security theorem [CPP25]

We say that a program G is secure against a **single-fault** attack with fault F if :



$$
\begin{array}{rl}
\text{if} & \text{initial-state } G\ S_0 \\
\text{and} & G \vDash_{\mathrm{F}} S_0 \xrightarrow{t}{}^{\star} S_3' \\
\text{and} & t = t_1 + [\text{Fault } F] \quad \text{and} \quad \text{nofault } t_1 \\
& \\
\text{then} & G \vDash_{\mathrm{F}} S_3' \xrightarrow{\epsilon}{}^{\star} \text{Caught} \\
\text{or} & \exists\ S_{n+1}\ t_2, \quad \text{nofault } t_2 \quad \text{and} \quad G \vDash_{\mathrm{F}} S_3' \xrightarrow{t_2}{}^{\star} S_{n+1} \quad \text{and} \quad G \vdash S_0 \xrightarrow{t_1 + t_2}{}^{\star} S_{n+1}
\end{array}
$$

Analysis in isolation of LM schemes



LAZART

Isolation analysis with *Data Load* and *Branch Inversion* fault models

- Input : the value stored in `%var` memory cell
- Output : the value loaded in `%target`
- Nominal behavior : `%target` stores `%var`'s value

Robustness levels of countermeasure schemes with limit=4

	Fault model					
Countermeasure	Test inv.		Load modif.		Comb	
Test duplication	1	2	4	0	1	2
Load duplication	4	0	1	1	1	2
Load triplication	4	0	2	1	2	4

Placement of software countermeasures [FDTC 23]

⇒ help to place countermeasures against multi-fault attacks.

A two steps approach :

1. **Isolation analysis** of protection schemes. Compute *robustness level* : *How many faults at least are required to produce an invalid behavior (not detected) ?*
2. **Placement algorithms**. Select the protection to apply to each IP in the program, using a representative set of attacks on the program wrt to a set of fault models M.

Table – Principle of each placement algorithms

Algorithm	Description
`naive`	All IPs in P are protected with $vl > N$
atk	All IPs in attacks are protected with $vl > N$
min	All IPs in minimal attacks are protected with $vl > N$
`block`	At least one IP per minimal attacks is protected with $vl > N$
`opt`	Protection is distributed between the IPs in minimal attacks, to get rid of attacks in less than $N+1$ faults.

Outline

State of the Art

× Temporary solutions : O0, programming tricks, optimization deactivation . . .

◇ Vu Son Tuan (21) : "preserving properties throughout the optimizing compilation flow"

```
void burn( void *v, size_t n )
{
  volatile unsigned char *p = ( volatile unsigned char * )v;
  while( n-- ) *p++ = 0;
}
```

Experimental result with CompCert (CPP 25)

Prog.	No CM, 00 #IP	1F	2F	CM, 00 #IP	1F	2F	CM, 01 #IP	1F	2F	CM+cpy, 01 #IP	1F	2F
vp	4	3	3	16	0	5	15	1	4	16	0	5
ark	1	1	0	4	0	2	3	1	0	4	0	2
aes	2	2	3	8	0	4	8	0	4	8	0	4
fu	11	4	13	41	0	5	23	3	1	34	0	3

- 01 is with optimization ; there are no higher optimization levels in CompCert
- cpy implements an opaque copy directive strengthened countermeasures against optimizations
  `__builtin_copy_##type((val, --LINE--)` : identity where two copies of the same value are differenciated.

The future

Open challenges :

- Combining countermeasures and fault models
  - Combinatory and Compositionality
- Preserving countermeasures without adaptating optimizations
  - resistance of optimizations against attacks
  - define semantic properties attached to countermeasures (?)

A General formal framework (Smonad) :

$$\overbrace{\mathsf{src} \xrightarrow[\substack{\mathsf{without\_IP} \\ \textit{defensive}}]{{}^{0}\gtrsim^{n}} \mathsf{src} \xrightarrow[\substack{\mathsf{protect} \\ \textit{contre-mesure} \\ \textit{+ insertion IP}}]{{}^{n}\gtrsim^{n}} \mathsf{protected} \xrightarrow[\substack{\mathsf{optim} \\ \textit{propagation} \\ \textit{de constantes}}]{{}^{n}\gtrsim^{n}} \mathsf{optimised}}^{\mathsf{Sem\_IP}} \xrightarrow[\mathsf{defensive}]{{}^{n}\gtrsim^{n}} \overbrace{\mathsf{optimised} \xrightarrow[\substack{\mathsf{remove\_IP} \\ \textit{suppression IP}}]{{}^{n}\gtrsim^{n}} \mathsf{trg}}^{\mathsf{Sem\_IP\_Brc}}$$

Raffinement : $src\ {}^{n}\gtrsim^{m}\ trg$

Publications

[CPP 25] Basile Pesin, Sylvain Boulmé, David Monniaux, Marie-Laure Potet. Formally Verified Hardening of C Programs against Hardware Fault Injection. 14th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP'25)
[FDTC 23] Etienne Boespflug, Laurent Mounier, Marie-Laure Potet, Abderrahmane Bouguern A compositional methodology to harden programs against multi-fault attacks Workshop on Fault Diagnosis and Tolerance in Cryptography, (FDTC 2023)

[JCEN 23] Guilhem Lacombe, David Féliot, Etienne Boespflug, Marie-Laure Potet. Combining static analysis and dynamic symbolic execution in a toolchain to detect fault injection vulnerabilities. JCEN, january 2023

[ASE 23] Soline Ducousso, Sébastien Bardin, Marie-Laure Potet. Adversarial Reachability for Program-level Security Analysis. European Symposium on Programming (AESE), april 2023

[FDTC 2020] Etienne Boespflug, Cristian Ene, Laurent Mounier, Marie-Laure Potet Countermeasures Optimization in Multiple Fault-Injection Context Workshop on Fault Diagnosis and Tolerance in Cryptography, (FDTC 2020)

[SAFECOMP 2016] Louis Dureuil, Guillaume Petiot, Marie-Laure Potet, Thanh-Ha Le, Aude Crohen, Philippe De Choudens. FISSC : a Fault Injection and Simulation Secure Collection. SAFECOMP 2016

[Cardis 2015] Louis Dureuil and Marie-Laure Potet and Philippe de Choudens and Cécile Dumas and Jessy Clédière. From Code Review to Fault Injection Attacks : Filling the Gap using Fault Model Inference. Cardis 2015

[ICST 2014] Marie-Laure Potet, Laurent Mounier, Maxime Puys and Louis Dureuil. Lazart : a symbolic approach to evaluate the impact of fault injections by test inverting. ICST 2014, International Conference on Software Testing

HDR durant JAIFs (- :

- Jessy Clédière (2013)
- Jean-Max Dutertre (2017)
- Karine Heydemann (2017)
- Jean-Baptiste Rigaud (novembre 21)
- Damien Couroussé (aout 24)
- Paolo Maistri (novembre 24)
- Ronan Lashermes (mai 25)
- Guillaume Bouffard (septembre 2025)
- ...

et de très nombreuses thèses (soutenues/en cours) sur de nombreux sujets !