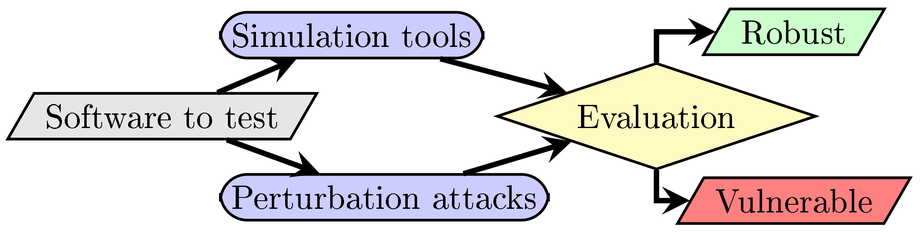
Evaluation of the Embedded Software Robustness Against Intentional Fault Injections by Simulation

Maël Berthier[2], Cécile Dumas[1], Louis Dureuil[1,3], Thanh-Ha Le[2], Marie-Laure Potet[3], Lionel Rivière[2]

[1]CEA-LETI
[2]SAFRAN MORPHO
[3]VERIMAG, University of Grenoble

SERTIF project

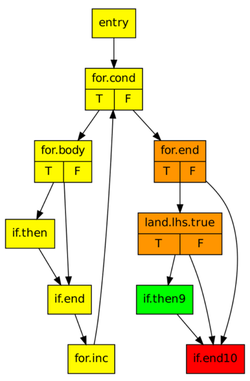
- Evaluate software implementations against fault injection attacks targeting data and control flow.
- Propose robustness evaluation criteria of software implementations.
- Compare the simulation tools independently developed by the partners.
- Build a benchmark of smartcard applications directed towards fault injection.

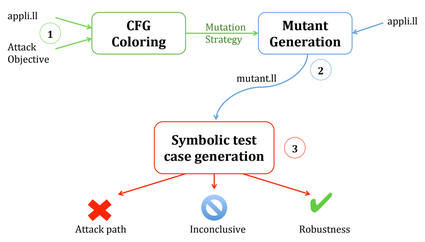


Fault simulators

Lazart (by VERIMAG)

- C code robustness evaluation against fault injection using symbolic execution.

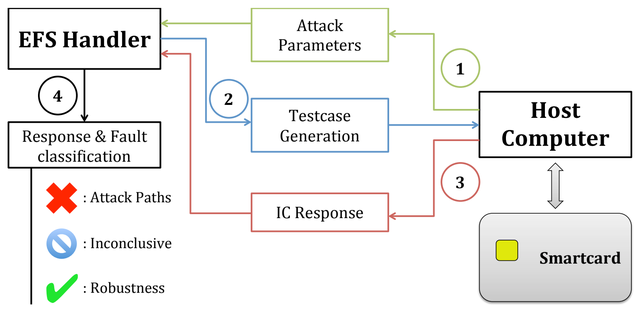




- Goal: Reach or avoid a CFG block.
- Fault model: control-flow condition inversion.
- Based on Klee, a concolic tool for LLVM.
- A complete diagnostic: activates all possible paths and fault injections.
- Scales to multiple fault injection scenarios.

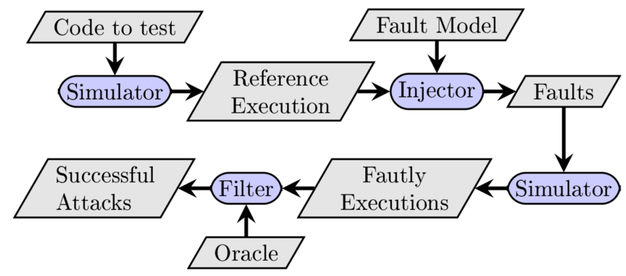
EFS (by MORPHO)

- **E**mbedded **F**ault **S**imulator: embedded into the target device (smartcard, micro-controller), at the low-level assembly code.
- Fault mechanism: a self-test program with a high priority level, granting access to critical registers, memories and execution flow of the smartcard.
- Fault models: code alterations (instruction skipping, instruction alteration), data modification at register level.
- Advantages:
  - fault injections on physical component.
  - side-channel observations.

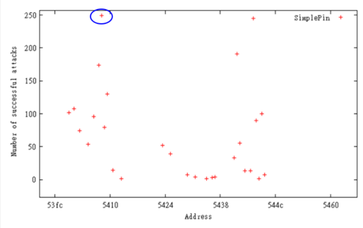


CELTIC (by CEA-LETI)

- Native smartcard binaries simulation with fault injection.



- Custom Domain Specific Language to decode and execute native instructions.
- Fault model: volatile memory perturbation, can model data and code faults.
- User-defined victory oracles.




```
...
0x540c: MOV A, R0
0x540d: CMP A, #04h Replace
0x540e: JNC D0
...
```

Combining high-level and low-level simulations

(Paper "Combining High-Level and Low-Level Approaches to Evaluate Software Implementations Robustness Against Multiple Fault Injection Attacks", FPS 2014)

- **Observation:** vulnerability sets detected by Lazart and EFS often intersect, however each simulator also detects vulnerabilities that are not revealed by the others tool.

Example *byteArrayCompare*

```
1  // Byte array comparison
2  static byte byteArrayCompare(byte* a1, byte* a2){
3    int i       = 0;
4    byte status  = BOOL_FALSE;
5    byte diff    = BOOL_FALSE;
6    for (i=0; i<pinSize; i++)
7        if (a1[i] != a2[i])
8            diff = BOOL_TRUE;
9    if ((i == pinSize) && (diff == BOOL_FALSE))
10       status = BOOL_TRUE;
11   return status;
12 }
```

Lazart		EFS	
Fault number	*Attacks*	*Skipped instructions*	*Attacks*
0	0	0	0
1	1	1	4
2	1	2	1
3	0	3	1
4	1	4+	0
Total	3	Total	6

Example *verifyPIN*

```
1  equal = BOOL_TRUE;
2  for(i=0 ; i<pinSize; i++) { // Main comparison
3      equal = equal & ((userPin[i] != cardPin[i]) ? BOOL_FALSE :
       BOOL_TRUE);
4      stepCounter++;
5  }
6  if(equal == BOOL_TRUE) {
7      if(equal != BOOL_TRUE)    // Double test
8          goto counter_measure;
9      ptc = MAX_TRIES;     // PIN Try counter (PTC) backup
10     ptcTst = -MAX_TRIES;     // Second backup for test
11     if(ptc != -ptcTst) // Verifies the new value
12         goto counter_measure;
13     authenticated = 1; // Authentication status update
14     if(stepCounter == INITIAL_VALUE + 4)
15         return EXIT_SUCCESS;
16 } else {
17     authenticated = 0;
18     if (stepCounter == INITIAL_VALUE + 4)
19         goto failure;
20 }
```

Lazart		EFS	
Fault number	*Attacks*	*Skipped instructions*	*Attacks*
0	0	0	0
1	0	1	1
2	2	2	1
3	0	3	0
4	1	4	0
		5+	1
Total	3	Total	3

- **Optimization:** combining the simulation tools revealed enhanced vulnerability detection, accuracy and coverage.

byteArrayCompare

Approach	*Tests*	*Attacks*	*Detection Rate*	*Time*
Lazart	56	27 (3)	11,7%	≈ 3s
EFS	2652	204 (6)	2,9%	≈ 9mn
Both	56 + 572	20 (4)	20%	≈ 2mn

verifyPIN

Approach	*Tests*	*Attacks*	*Detection Rate*	*Time*
Lazart	49	18 (3)	16,6%	< 3s
EFS	4528	72 (2)	2,7%	≈ 17mn
Both	49 + 720	14 (3)	21.4%	≈ 1.5mn

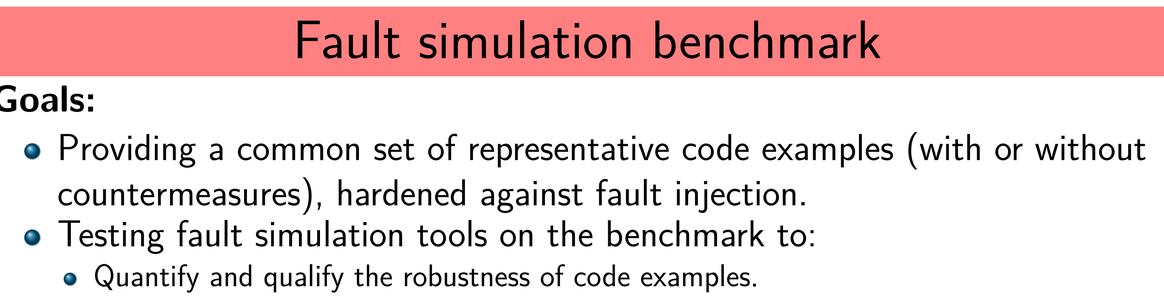
Fault simulation benchmark

Goals:

- Providing a common set of representative code examples (with or without countermeasures), hardened against fault injection.
- Testing fault simulation tools on the benchmark to:
  - Quantify and qualify the robustness of code examples.
  - Establish relevant comparisons between the tools.

Organization:

- Two categories of examples:
  - Code snippets to evaluate tools and their fault models.
  - Full implementations, to qualify their relative robustness.
- For each code example, we provide:
  - Source code (in C).
  - Victory oracle (conditions for an attack to be successful).
  - Toolchain (OS, compiler) and compilation invocation.
  - Relevant information about the expected memory layout.



Perspectives of SERTIF

- Extension to secure elements or smart secure devices.
- Robustness against high-order fault injection.
- Studies of compiler impact on robustness and counter-measures.

SERTIF: ANR-14-ASTR-0003-01 Simulation for the Evaluation of Robustness of embedded Applications against Fault injection.