Fault Injection Attacks Against an In-Core DIFT Mechanism

Cyberus Summer School 2023

William PENSEC

Université Bretagne Sud
Lab-STICC

July 3, 2023

Context

- Rapid expansion of connected objects.
- Increased attack surface.
- Objects with physical proximity and network connectivity.
- Software and physical threats.

Issue

- How can we maintain maximum protection against software attacks in the presence of physical attacks?

Outline

Information Flow Tracking : what is it ?

IFT overview

- 2 categories: static or **dynamic**
- Different types: software, **hardware**, hybrids [3]
- Protection against software attacks (e.g.: *buffer overflow*, *format string*, *SQL injections*, ...) [2, 5]

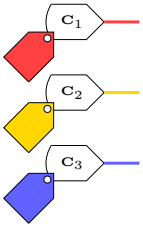
DIFT principle

- We attach **labels** called tags to **containers** and specify an information flow **policy**, i.e. relations between tags
- At runtime, we **propagate** tags to reflect information flows that occur and **detect** any **policy violation**

DIFT : how does it work ?

Three steps

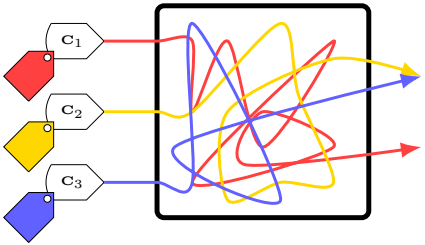
- Tag initialization



DIFT : how does it work ?

Three steps

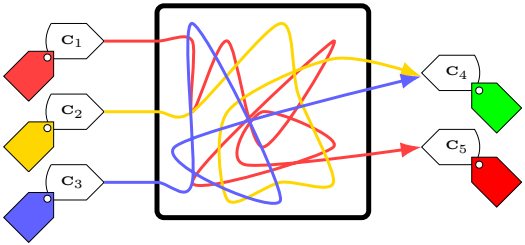
- Tag initialization
- Tag propagation



DIFT : how does it work ?

Three steps

- Tag initialization
- Tag propagation
- Tag check



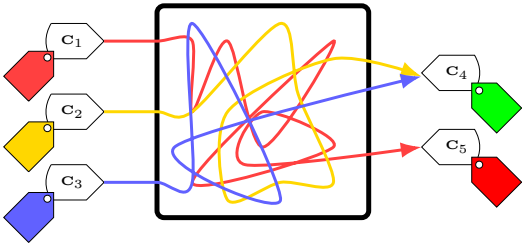
DIFT : how does it work ?

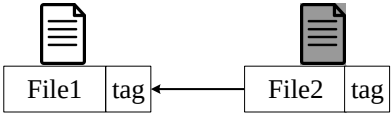
Three steps

- Tag initialization
- Tag propagation
- Tag check

Levels of IFT

- Application level





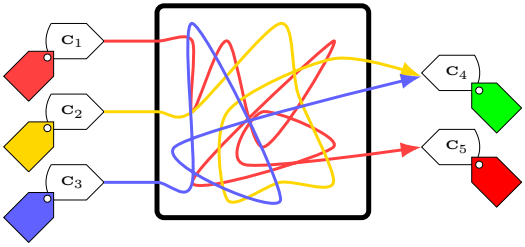
DIFT : how does it work ?

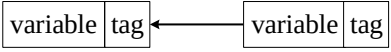
Three steps

- Tag initialization
- Tag propagation
- Tag check

Levels of IFT

- Application level
- OS level





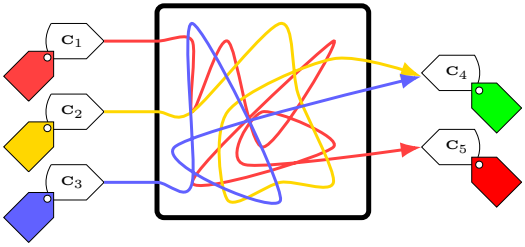
DIFT : how does it work ?

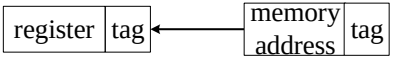
Three steps

- Tag initialization
- Tag propagation
- Tag check

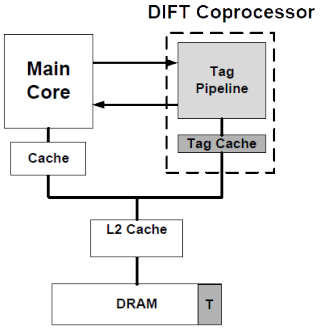
Levels of IFT

- Application level
- OS level
- Low level





Off-core DIFT



Co-processor dedicated to DIFT [6, 11]

Advantages

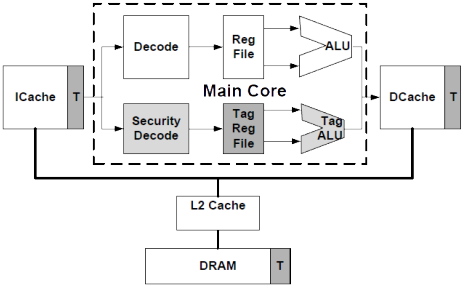
Flexible security policy
Low overhead (<10 %)
Unmodified CPU

Disadvantages

Communication between CPU and DIFT co-processor

Type of DIFT used in the Hardblare project developed at Lab-STICC [1]

In-core DIFT



	Advantages	Disadvantages
DIFT in-core [4, 9]	Low overhead (<10%)	Invasive changes Few security policies

Few security policies, but little impact on many embedded systems

Outline

Motivations

- FIA can be performed by disturbing the power supply or the clock, by using EM pulses or laser shots [7].
- The impact of an injection varies depending on the type of FIA.
- Many studies have shown the vulnerabilities of critical systems against FIA :
  - Glitch injections : Voltage glitches can lead to glitch trust-zone mechanisms as shown in [10], power supply to control the program counter [12],
  - EM Fault Injection (EMFI) : to recover an AES key by targeting the cache hierarchy and the MMU as shown in [13],
  - SCA/FIA : [8] have shown that you can combine side-channel attacks (SCA) and FIAs to bypass the PMP mechanism in a RISC-V processor

In this work

- We propose to study the combination of software and physical attacks to defeat the DIFT mechanism implemented in the D-RI5CY processor.

D-RI5CY

- Design [9] made by researchers at Columbia University (USA) in partnership with the University of Turin (Italy).
- Based on the 32-bit RISC-V processor: RI5CY (PULP platform).
- Tags on 1 bit in the core, but 4 bits in memory.
- Flexible security policy that can be modified while an application is running.

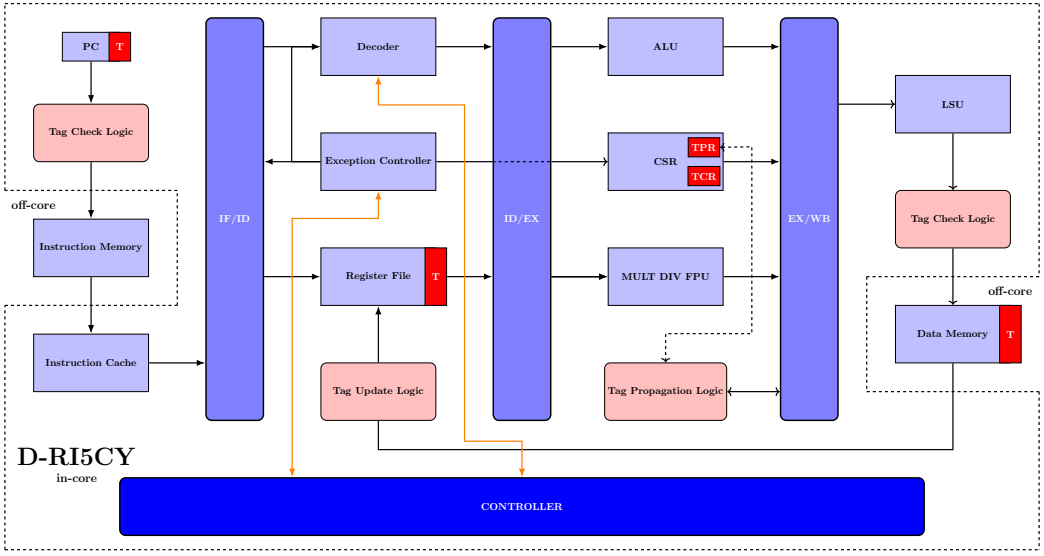


Figure 1: Architecture of the D-RI5CY. DIFT components in red and pink

Threat model

We consider an attacker able to

- combine software and physical attacks to defeat the DIFT mechanism
- To ensure an exhaustive campaign, we inject faults in registers associated to the DIFT-related components
  - set to 0,
  - set to 1,
  - a bit-flip in all position of the targeted register,
  - keep the value of the previous cycle

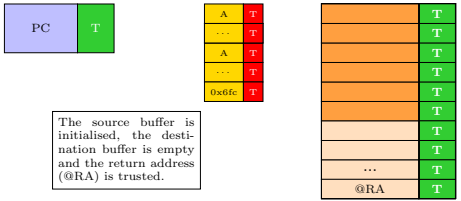
Outline

Vulnerability Assessment

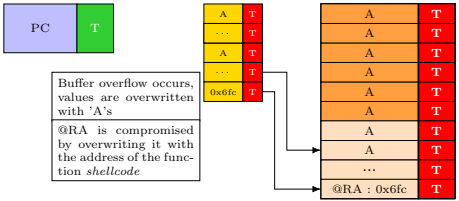
- Analysis of 4 cases: buffer overflow, format string, compare/store, compare/compute.
- In-depth study of these cases.
- Analysis of tag propagation temporally and logically.
- Presentation of only 2 cases in this presentation

Case 1: Buffer overflow

- The attacker exploits a buffer overflow to access the return address register (*ra*).



(a) Malicious buffer and *ra* trusted



(b) Overflow and overwriting of *ra* and its tag

- Thanks to DIFT, the tags associated with the buffer data overwrite the *ra* register tag.
- As the data in the buffer is manipulated by the user, it is marked as *untrusted*.
- When the function returns, the corrupted register *ra* is loaded into *PC* using a jalr instruction.

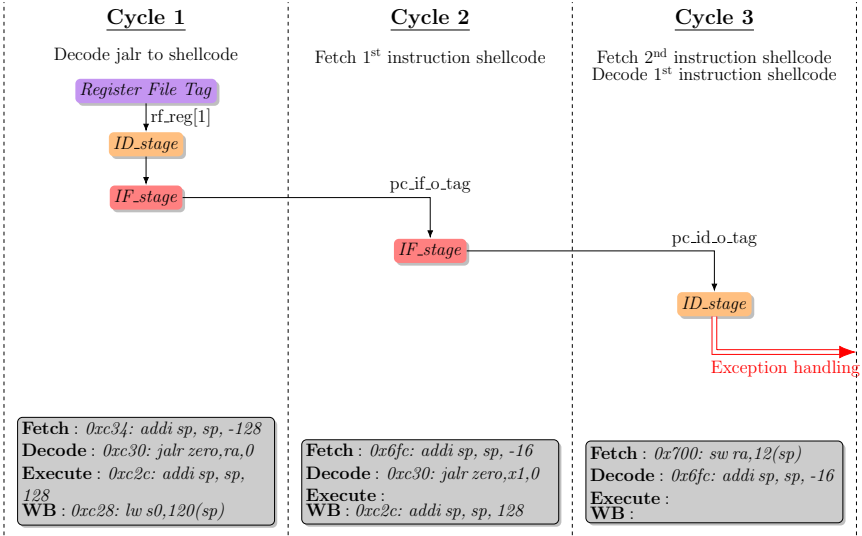


Figure 3: Temporal analysis of the tags propagation in *Buffer Overflow* attack

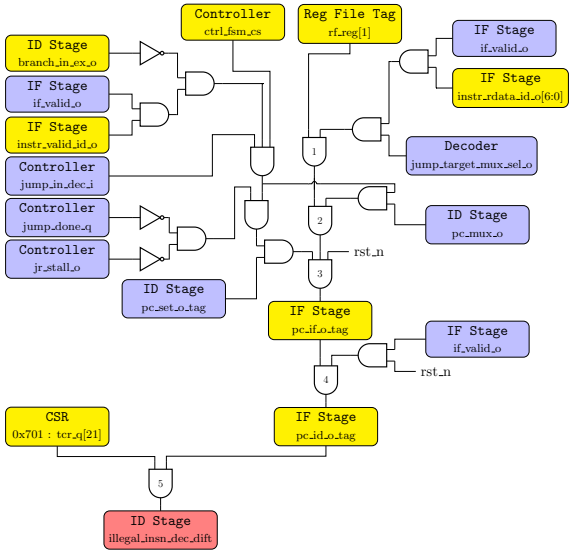


Figure 4: Logical analysis of the tags propagation in a *Buffer Overflow* attack

Case 2: WU-FTPd

- The vulnerability is the use of an unchecked user input as the format string parameter in functions that perform formatting, e.g. printf()
- An attacker can use the format tokens, to write into arbitrary locations of memory, e.g. the return address of the function.

```
void echo(){
    int a;
    register int i asm("x8");
    a = i; // &a = 107FD0
    printf("%224u%n%35u%n%253u%n%n", 1, (int*) (a-4), 1, (int*)
}
```

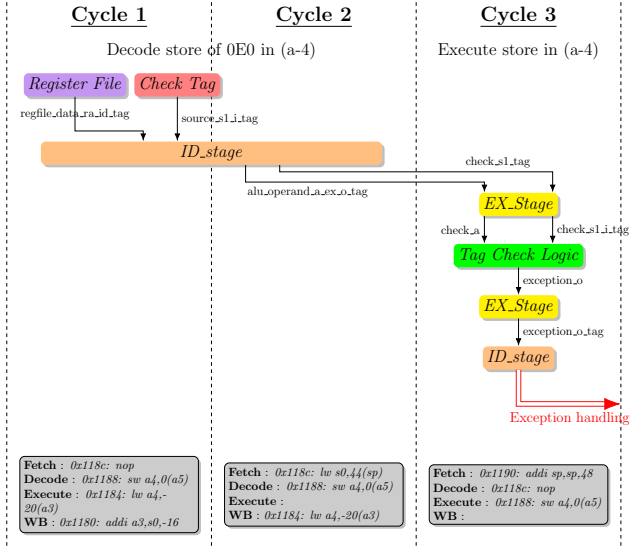


Figure 5: Temporal analysis of the tags propagation in a *format string* attack

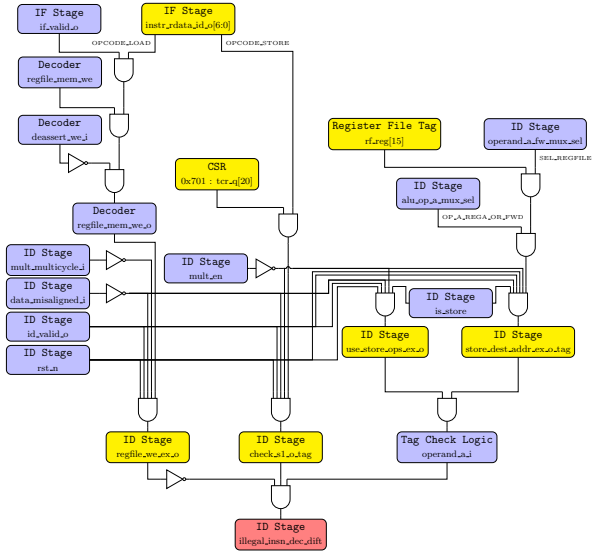


Figure 6: Logical analysis of the tags propagation in a *format string* attack

Outline

Experimental Setup - Simulation fault injections campaign

- Logical fault injection simulation is used for preliminary evaluations
  - faults are injected in the HDL code at cycle accurate and bit accurate level
  - a set of 54 DIFT-related registers are targeted
  - a set of attack windows are determined based on the previous study
  - results are classed in four groups
    - crash: reference cycle count exceeded,
    - Nothing Significant To Report (NSTR)
    - delay: illegal instruction is delayed
    - success: DIFT has been bypassed
- Simulations with QuestaSim 10.6e.

Outline

Main results : 4 cases

Table 1: End of simulation status

	Crash	NSTR	Delay	Success	Total
Buffer overflow	0	1677	25	26 (1.50%)	1728
WU-FTPd	0	1886	79	51 (2.53%)	2016
Compare/Store	0	1657	58	13 (0.75%)	1728
Compare/Sub/Add	0	1113	12	27 (2.34%)	1152

Buffer overflow

Table 2: Buffer overflow : Register sensitivity as determined by fault model and simulation time

	137140 ns				137180 ns				137220 ns				137260 ns				137300 ns			
	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay
pc_if_o_tag								✓	✓			✓	✓		✓					
rf_reg[1]								✓	✓		✓									
tcr_q	✓		✓		✓		✓		✓		✓		✓		✓		✓		✓	
tpr_q	✓	✓	✓		✓	✓	✓													

Table 3: WU-FTPd : Register sensitivity as determined by fault model and simulation time

	2099100 ns				2099140 ns				2099180 ns				2099220 ns				2099260 ns				2099300 ns				2099340 ns				2099380 ns			
	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay	set0	set1	bitflip	delay
alu_operand_b_ex_o_tag				✓	✓		✓																									
alu_operator_o_mode					✓	✓	✓																									
check_s1_o_tag																								✓	✓			✓	✓		✓	
regfile_alu_waddr_ex_o_tag																			✓													
rf_reg[15]																					✓		✓		✓		✓					
store_dest_addr_ex_o_tag																								✓	✓			✓	✓		✓	
tcr_q					✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		✓		✓					
tpr_q						✓	✓			✓	✓			✓	✓			✓	✓													
use_store_ops_ex_o																								✓	✓			✓	✓		✓	

Outline

Conclusion

- We have shown that the D-RI5CY DIFT mechanism is vulnerable to FIAs
- We identified 13 DIFT-related sensitive registers
- 117 simulated fault injections over 6624 have lead to a successful attack (1.77%)
- 2.63% of the simulated injections delay the DIFT exception

Perspectives

- In future works we will
  - Extend the D-RI5CY DIFT mechanism with countermeasures to face fault injection attacks
  - Extend our study to the entire D-RI5CY core since several processor registers can also impact the robustness of the DIFT mechanism against FIA
  - Strengthen the proposed analysis through actual fault injection campaign targeting a FPGA implementation

Thank you for your attention.

If you have any questions, feel free to ask them now.

References

[1] Mounir Nasr Allah et al. “Hardblare: a hardware-assisted approach for dynamic information flow tracking”. In: *Séminaire des doctorantes et doctorants en informatique de la Société Informatique de France*. 2016.

[2] Christopher Brant et al. “Challenges and Opportunities for Practical and Effective Dynamic Information Flow Tracking”. In: *ACM Computing Surveys (CSUR)* (2021). DOI: 10.1145/3483790.

[3] Kejun Chen et al. “Dynamic Information Flow Tracking: Taxonomy, Challenges, and Opportunities”. In: *Micromachines* (2021). DOI: 10.3390/mi12080898.

[4] Michael Dalton, Hari Kannan, and Christos Kozyrakis. “Raksha: A Flexible Information Flow Architecture for Software Security”. In: *Proceedings of the 34th Annual International Symposium on Computer Architecture*. Association for Computing Machinery, 2007. DOI: 10.1145/1250662.1250722.

[5] Wei Hu, Armaiti Ardeshiricham, and Ryan Kastner. “Hardware Information Flow Tracking”. In: *ACM Computing Surveys* (2021). DOI: 10.1145/3447867.

References

[6] Hari Kannan, Michael Dalton, and Christos Kozyrakis. "Decoupling Dynamic Information Flow Tracking with a Dedicated Coprocessor". In: *International Conference on Dependable Systems & Networks*. IEEE, 2009. DOI: 10.1109/DSN.2009.5270347.

[7] Duško Karaklajić, Jörn-Marc Schmidt, and Ingrid Verbauwhede. "Hardware Designer's Guide to Fault Attacks". In: *IEEE Transactions on Very Large Scale Integration Systems* (2013). DOI: 10.1109/TVLSI.2012.2231707.

[8] Shoei Nashimoto et al. "Bypassing Isolated Execution on RISC-V using Side-Channel-Assisted Fault-Injection and Its Countermeasure". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)* (2021). DOI: 10.46586/tches.v2022.i1.28-68.

[9] Christian Palmiero et al. "Design and Implementation of a Dynamic Information Flow Tracking Architecture to Secure a RISC-V Core for IoT Applications". In: *2018 IEEE High Performance extreme Computing Conference (HPEC)*. 2018, pp. 1–7. DOI: 10.1109/HPEC.2018.8547578.

[10] Marvin Saß, Richard Mitev, and Ahmad-Reza Sadeghi. "Oops..! I Glitched It Again! How to Multi-Glitch the Glitching-Protections on ARM TrustZone-M". In: 2023. DOI: 10.48550/arXiv.2302.06932. arXiv: 2302.06932 [cs.CR].

References

[11] G. Edward Suh et al. “Secure Program Execution via Dynamic Information Flow Tracking”. In: *SIGPLAN Not.* (2004). DOI: 10.1145/1037187.1024404.

[12] Niek Timmers, Albert Spruyt, and Marc Witteman. “Controlling PC on ARM Using Fault Injection”. In: *Fault Diagnosis and Tolerance in Cryptography (FDTC)*. 2016. DOI: 10.1109/FDTC.2016.18.

[13] Thomas Trouchkine et al. “Electromagnetic Fault Injection Against a Complex CPU, toward new Micro-architectural Fault Models”. In: *Journal of Cryptographic Engineering* (2021). DOI: 10.1007/s13389-021-00259-6.