HardBlare: Monitoring information flows in heterogeneous SoCs with a dedicated coprocessor

Muhammad Abdul Wahab [α], Pascal Cotret [λ], Mounir Nasr Allah [β], Guillaume Hiet [β], Vianney Lapôtre [γ], Guy Gogniat [γ], Arnab Kumar Biswas [γ]

[α] SCEE / IETR, CentraleSupélec, Cesson-Sévigné - FRANCE, [β] CIDRE / IRISA / INRIA, CentraleSupélec, Cesson-Sévigné - FRANCE, [γ] Lab-STICC, University of South Brittany, Lorient - FRANCE, [λ] Chercheur indépendant - FRANCE

Introduction

HardBlare proposes a software/hardware co-design methodology to ensure that security properties are preserved all along the execution of the system but also during files storage. The general context is to address **Dynamic Information Flow Tracking (DIFT)** that generally consists in attaching marks (also known as tags) to denote the type of information that are saved or generated within the system. It allows to detect software attacks such as **overflows** or **SQL injection**.
Example: Let's suppose that "print" function, which sends output stream to stdout, is public and the tag of a variable `x` is underlined variable <u>x</u>.

Example code	Tag initialization	Tag propagation	Tag check
`p = 3;`	<u>p</u> ← public		
`s = 42;`	<u>s</u> ← secret		
`x = p + s;`		<u>x</u> ← <u>p</u> + <u>s</u> = <u>s</u>	
`print(x);`			if (<u>x</u> != public) `raise interruption`

State of the art

		Advantages	Disadvantages
	Software	Flexible security policies	Huge runtime overhead (from 300% to 3700%)
Hybrid	In-core DIFT [1]	Low runtime overhead (<10%)	Invasive modifications Few security policies
Hybrid	Dedicated CPU for DIFT [2]	Low runtime overhead (<10%) Few modifications to CPU	Wasting resources Energy consumption (x 2)
Hybrid	Dedicated DIFT Coprocessor [3],[4],[5]	Flexible security policies Low runtime overhead (<10%) CPU not modified	Communication between CPU and DIFT Coprocessor

Recovering required information for DIFT on ARM hardcore CPU

1. CoreSight components allow to reconstruct the traced program's Control Flow Graph (CFG).
2. Static analysis allows to determine what happens inside each basic block.
3. Memory instructions (`ldr`, str, ...) are instrumented in order to retrieve memory addresses that cannot be recovered from static analysis. The register **r9 is dedicated** to the instrumentation and contain the address of the **instrumentation FIFO**.
4. RFBlare, a modified Linux kernel, provide support to store and retrieve tags associated to files.

```
0x10168 movw r0,#0x913c
        str sp, [r9]
        str r0, [sp, #4]
        str r2, [r9]
        ldr r1, [r2], #4
        bxls lr
```

Overall Architecture

- **User application** is executed on the **Cortex-A9**.
- When the Linux kernel loads the application, it also loads the output of static analysis in the **Annotations** memory section.
- During the execution, **PTM** sends to the **DIFT Monitor** the **traces** through the **PFT Decoder**.
- When the application performs memory accesses, it also sends the `load`/`store` addresses to the **DIFT Monitor** via the **Instrumentation FIFO**.
- If the application makes a `read` (`write`) syscall, **RFBlare** sends (retrieves) the tag to (from) the DIFT monitor.

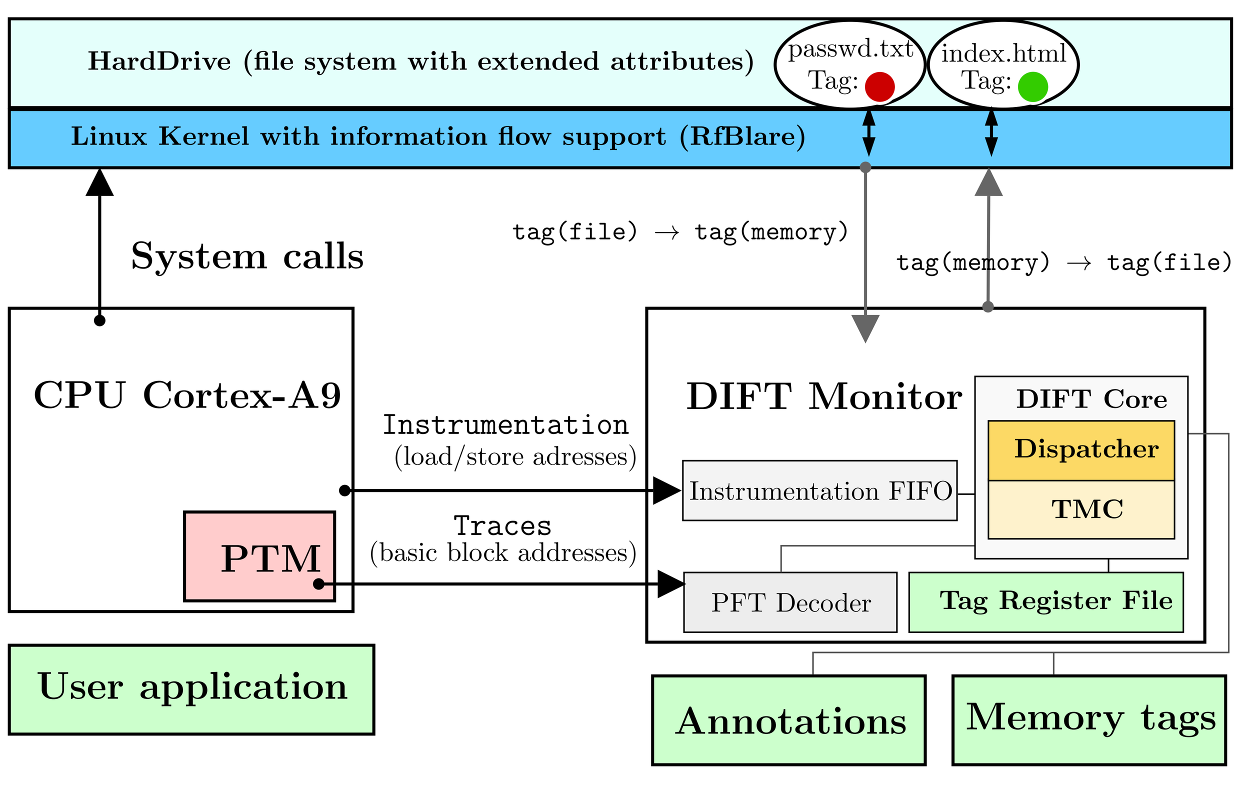

Figure: Overall architecture of proposed approach

- The DIFT monitor is implemented on the FPGA part of the Zynq SoC (ZedBoard).
- Using decoded trace, the **Dispatcher** core finds the corresponding **annotations** and store it in the memory for **TMC (Tag Management Core)**.
- DIFT Core
  - looks for tags either in **memory tags** or **tag register file**.
  - computes tags depending on propagation rules.
  - updates corresponding tags either in **memory tags** or **tag register file**.
  - checks for security policy violation and raise an interruption.

Implementation details and results

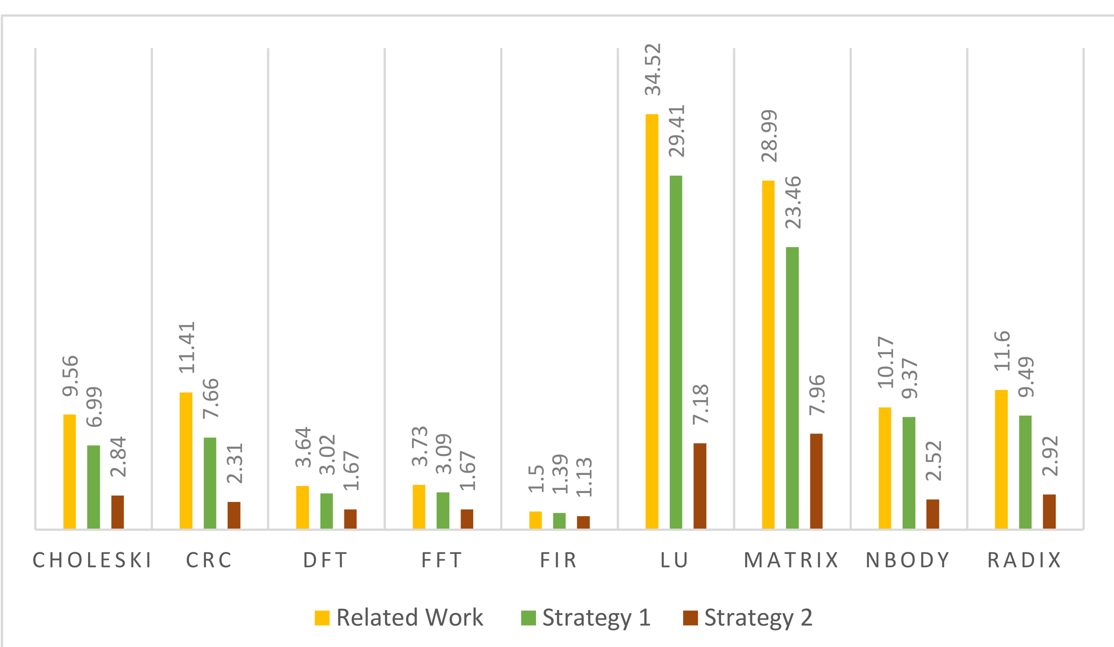

Figure: Average instrumentation time overhead

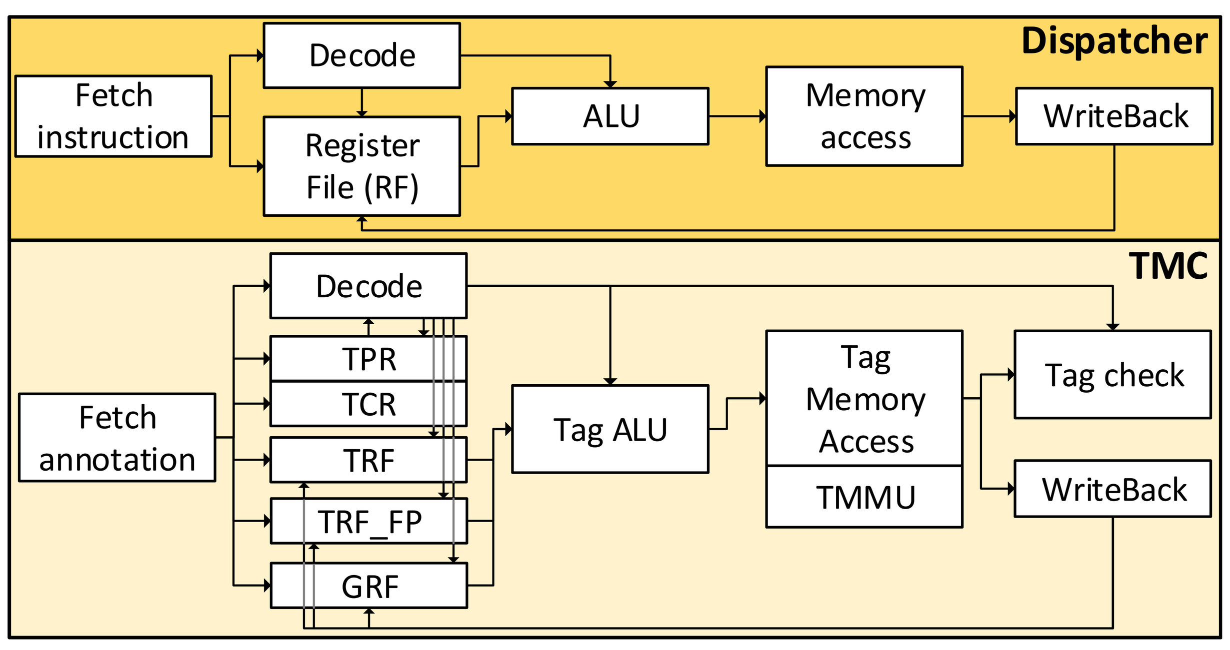

Figure: DIFT Core microarchitecture

Approaches	Kannan [3]	Deng [4]	Heo [5]	Wahab [6]	This work
Hardcore portability	No	No	Yes	Yes	Yes
Main CPU	Softcore	Softcore	Softcore	Hardcore	Hardcore
Communication overhead	N/A	N/A	60%	5.4%	335%
Library instrumentation	N/A	N/A	partial	No	Yes
All information flows	Yes	Yes	No	No	Yes
Area overhead	6.4%	14.8%	14.47%	0.47%	0.95 %
Power overhead	N/A	6.3%	24%	16%	16.2%
Max frequency	N/A	256 MHz	N/A	250 MHz	250 MHz
FP support	No	No	No	No	Yes
Multi-threaded support	No	No	No	No	Yes

Table: Performance comparison with related work

Some References

[1] M. Dalton, H. Kannan, and C. Kozyrakis, "Raksha: A flexible information flow architecture for software security," in *In International Symposium on Computer Architecture (ISCA)*, 2007.

[2] V. Nagarajan, H.-S. Kim, Y. Wu, and R. Gupta, "Dynamic information tracking on multicores," in *12th Workshop on the Interaction between Compilers and Computer Architecture (INTERACT)*, Feb 2008.

[3] H. Kannan, M. Dalton, and C. Kozyrakis, "Decoupling dynamic information flow tracking with a dedicated coprocessor," in *DSN 2009*, pp. 105–114, 2009.

[4] D. Y. Deng, D. Lo, G. Malysa, S. Schneider, and G. E. Suh, "Flexible and efficient instruction-grained run-time monitoring using on-chip reconfigurable fabric," MICRO '43, 2010.

[5] I. Heo, M. Kim, Y. Lee, C. Choi, J. Lee, B. B. Kang, and Y. Paek, "Implementing an application-specific instruction-set processor for system-level dynamic program analysis engines," *ACM TODAES*, 2015.

[6] M. A. Wahab, P. Cotret, M. N. Allah, G. Hiet, V. Lapotre, and G. Gogniat, "Armhex: A hardware extension for dift on arm-based socs," in *FPL 2017*, Sept 2017.

Conclusions and Perspectives

- Hardware-assisted DIFT system.
- Only solution to target ARM hardcore CPUs.
- Instrumentation overhead reduced by more than 3 times.
- Approach based on a non-modified CPU with a standard Linux and generic binaries ⇒ Could be implemented by industrial partners in medium-term.
- Perspectives on runtime reconfiguration and multicore/manycore systems.
- **Acknowledgements:** This work is done in the frame of **HardBlare** project which is funded by **CominLabs** and **Brittany** region.