HardBlare: a Hardware-Assisted Approach for Dynamic Information Flow Tracking

Mounir Nasr Allah, Guillaume Hiet, Muhammad Abdul Wahab, Pascal Cotret, Guy Gogniat, Vianney Lapotre

To cite this version:

Mounir Nasr Allah, Guillaume Hiet, Muhammad Abdul Wahab, Pascal Cotret, Guy Gogniat, et al.. HardBlare: a Hardware-Assisted Approach for Dynamic Information Flow Tracking. Séminaire des doctorantes et doctorants en informatique de la Société Informatique de France, Apr 2016, Paris, France. 2016. hal-01311032

HAL Id: hal-01311032

https://centralesupelec.hal.science/hal-01311032

Submitted on 23 Jun 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HardBlare: a Hardware-Assisted Approach for Dynamic Information Flow Tracking

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

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

Introduction

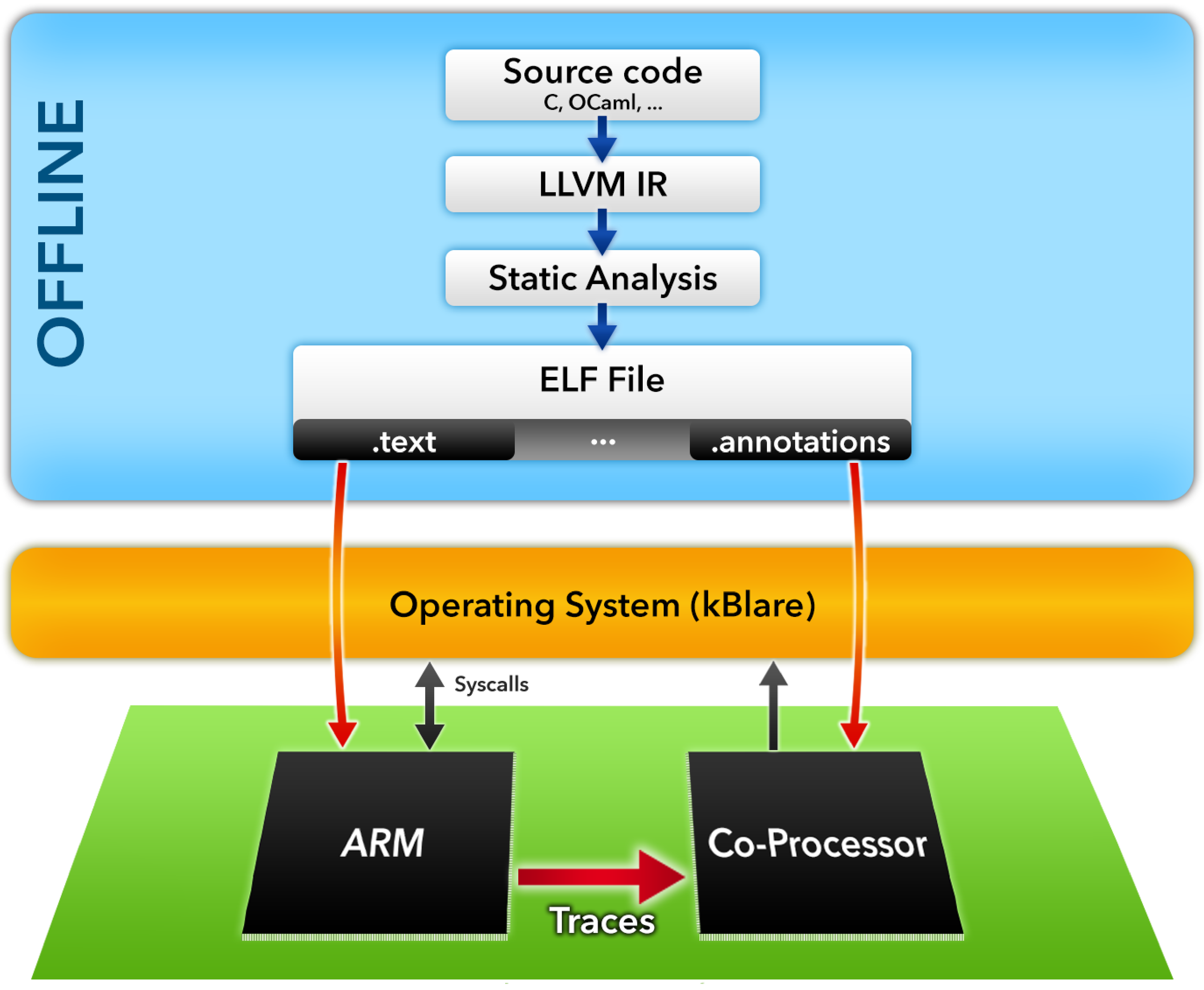
HardBlare proposes a software/hardware codesign methodology to ensure that security properties are preserved all along the execution of the system but also during files storage. The general context is to address **Dynamic Information Flow Tracking (DIFT)** that generally consists in attaching marks (also known as tags) to denote the type of information that are saved or generated within the system.
Let's suppose that "print" function is public and the tag of a variable x is underlined variable $\underline{x}$.

Example code	Tag initialization	Tag propagation	Tag check
`p = 3;`	$\underline{p} \leftarrow$ public		
`s = 42;`	$\underline{s} \leftarrow$ secret		
`x = p + s;`		$\underline{x} \leftarrow \underline{p} + \underline{s} = \underline{s}$	
`print(x);`			if ($\underline{x}$!= public) `raise interruption`

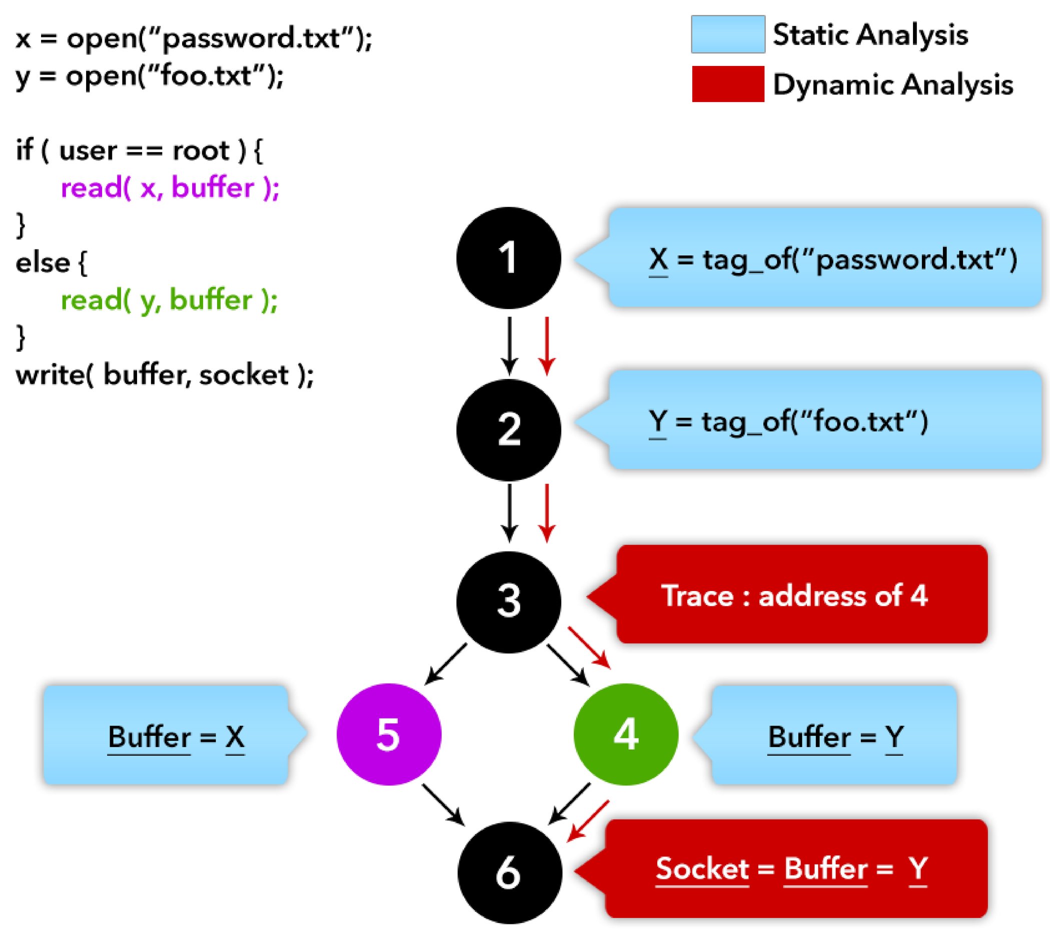
State of the art

		Advantages	Disadvantages
	Software	Flexible security policies Multiple attacks detected	Overhead (from 300% to 3700%)
	Hardware	Low overhead (<10%) Invasive modifications	Fixed Security policies
Hybrid	In-core DIFT	Low overhead (<10%) Few security policies	Invasive modifications
Hybrid	Dedicated CPU for DIFT	Low overhead (<10%) Few modifications to CPU	Wasting resources Energy consumption (x 2)
Hybrid	Dedicated DIFT Coprocessor	Flexible security policies Low overhead (<10%) CPU not modified	Communication between CPU and DIFT Coprocessor

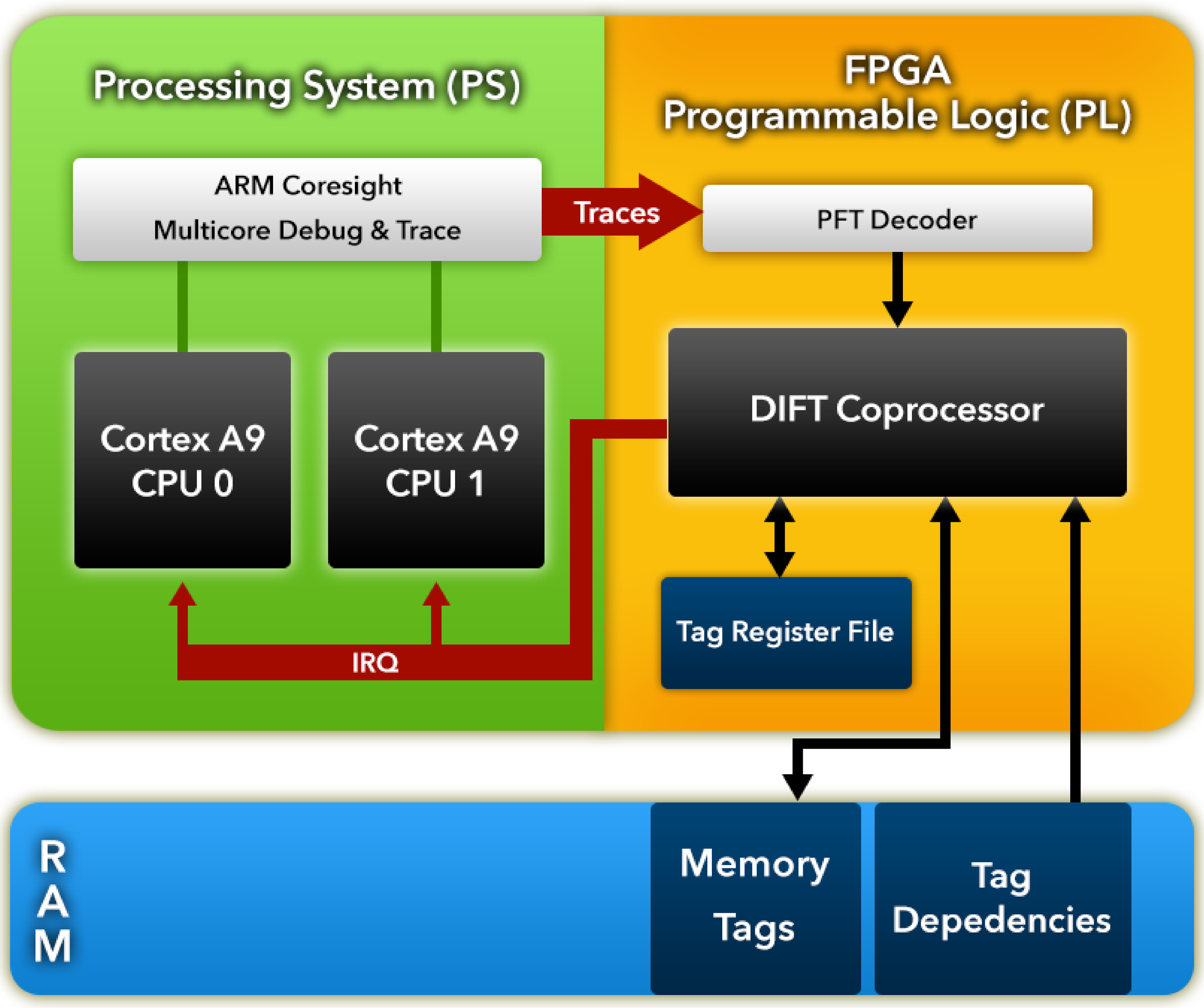
Static Analysis



- During the **compilation phase**, a **static analysis** is done on the **LLVM intermediate representation** produced from the source code, and propagated to the **ARM backend** for the machine code generation
- The **result of static analysis** gives a list of **dependencies** between information containers (e.g. registers, memory spaces...) for every **basic blocks** which are stored on a dedicated section in a **ELF File**
- During **run-time**, the **Program Trace Macrocell (PTM)** generates a **trace** containing the **address** for each committed instruction **modifying the PC value**
- **Annotations** related to the **basic block** identified by its address, given by the trace, are **processed by the coprocessor** to propagate tags



ARM Cortex-A9 Trace mode: Coresight components



Definitions

- **Tag dependencies** block contains annotations loaded when the program is launched
- **Memory tags** block contains tags related to information containers
- **Tag register file** contains tags related to CPU registers

DIFT step-by-step

- ARM CoreSight Components export trace (for both CPUs) towards PL in PFT **(Program Flow Trace)** protocol
- PFT Decoder decodes trace in usable format
- Using decoded trace, DIFT Coprocessor reads tag dependencies block
- DIFT Coprocessor looks for the tags either in memory or tag register file
- DIFT Coprocessor computes tags depending on propagation rules
- DIFT Coprocessor updates corresponding tags
- DIFT Coprocessor checks for security policy violation and raise an interruption

Some References

[1] G. Venkataramani, I. Doudalis, Y. Solihin, and M. Prvulovic, "Flexitaint: A programmable accelerator for dynamic taint propagation," in *In 14th International Symposium on HighPerformance Computer Architecture (HPCA-14*, 2008.

[2] 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.

[3] H. Kannan, M. Dalton, and C. Kozyrakis, "Decoupling dynamic information flow tracking with a dedicated coprocessor," in *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2009, Estoril, Lisbon, Portugal, June 29 - July 2, 2009*, pp. 105–114, 2009.

[4] 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.

[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 Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 4, p. 53, 2015.

Main Contributions at a Glance

- Hardware-assisted DIFT system with limited time overheads.
- Approach based on a non-modified CPU with a standard Linux and generic binaries $\Rightarrow$ Could be implemented by industrial partners in medium-term.
- Hardened with hardware security mechanisms: trusted coprocessor storage and bus protection in terms of confidentiality/integrity.
- Contributions on software-related issues as well (static/dynamic IFC analysis, i.e. hybrid analysis).
- Perspectives on runtime reconfiguration and multicore/manycore systems.

{mounir.nasrallah,muhammadabdul.wahab}@centralesupelec.fr