TrustGW : Hardware Dynamic Information Flow Tracking for FPGA-accelerated Applications

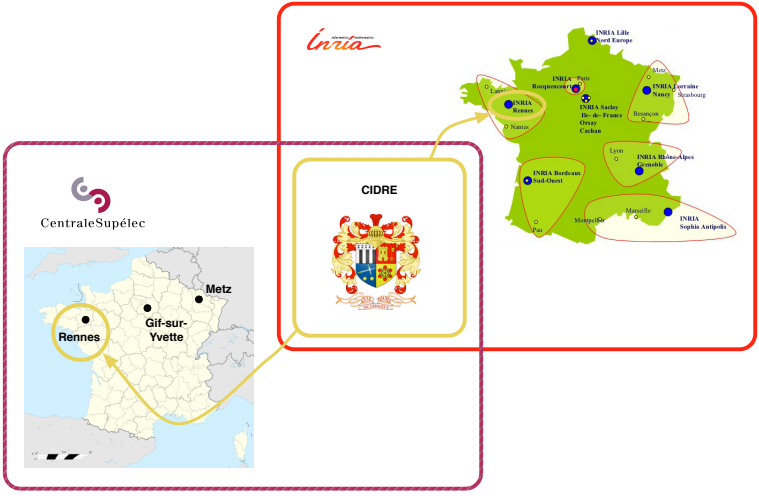
COEMS Forsterk Seminar 2022

Guillaume Hiet

CentraleSupélec, IRISA, CIDRE Inria Project-Team

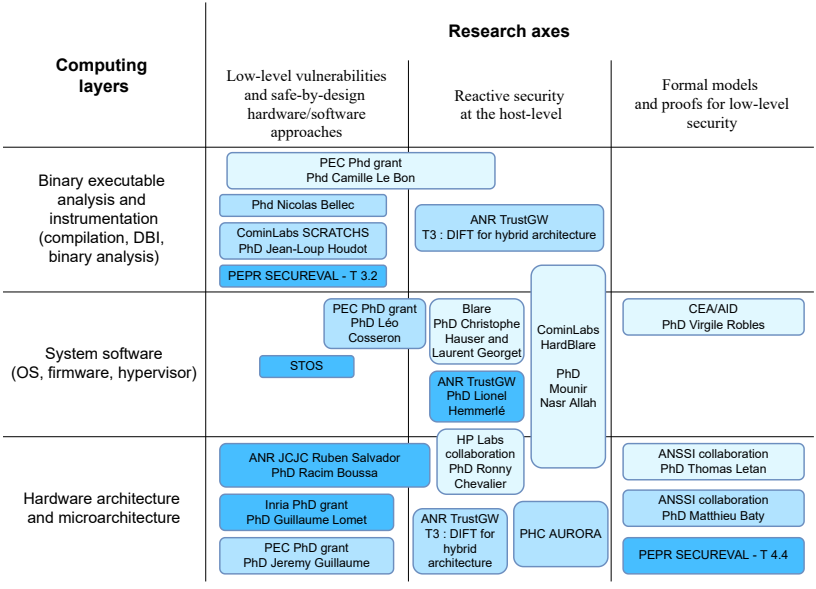
1. Dec. 2022

CIDRE: a joint team between CentraleSupelec and Inria



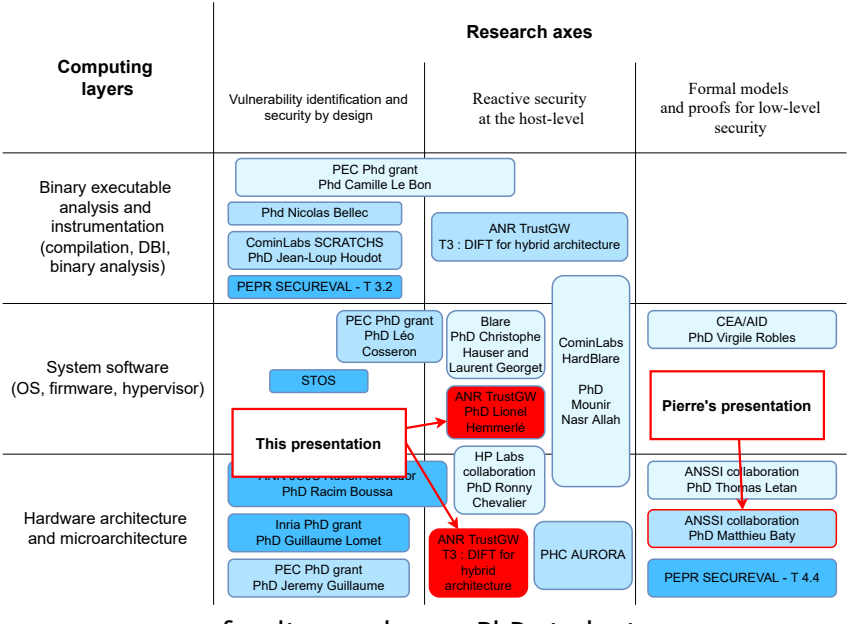
A research team of the IRISA laboratory: `https://www.irisa.fr/`

A new team on the security of hardware/software interfaces



5 faculty members, 7 PhD students

A new team on the security of hardware/software interfaces



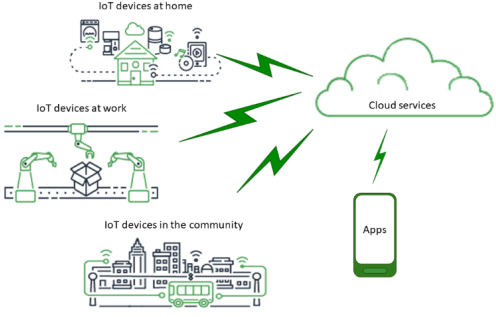
5 faculty members, 7 PhD students

Context of the TrustGW project

Cyber-security is a major concern

- Many vulnerable systems are targeted by sophisticated attacks
- Strongly connected to underground economy and military/intelligence activities

A new type of target: Embedded communication systems (e.g. IoT)



- Increasingly widespread in critical infrastructures
- They handle sensitive data
- They increase the global attack surface of information systems

Context of the TrustGW project

Cyber-security is a major concern

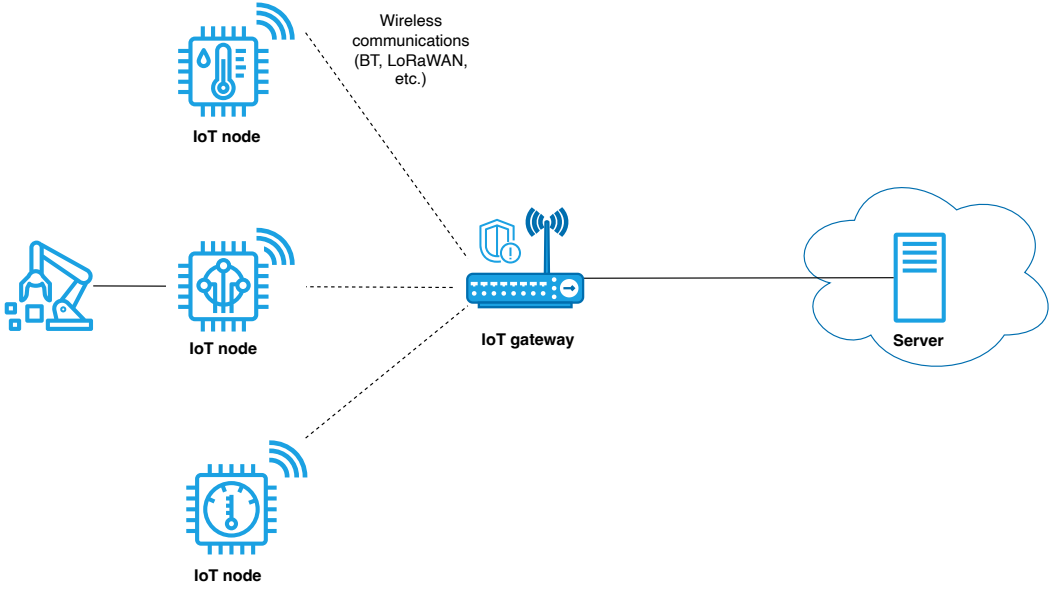
- Many vulnerable systems are targeted by sophisticated attacks
- Strongly connected to underground economy and military/intelligence activities

A new type of target: Embedded communication systems (e.g. IoT)

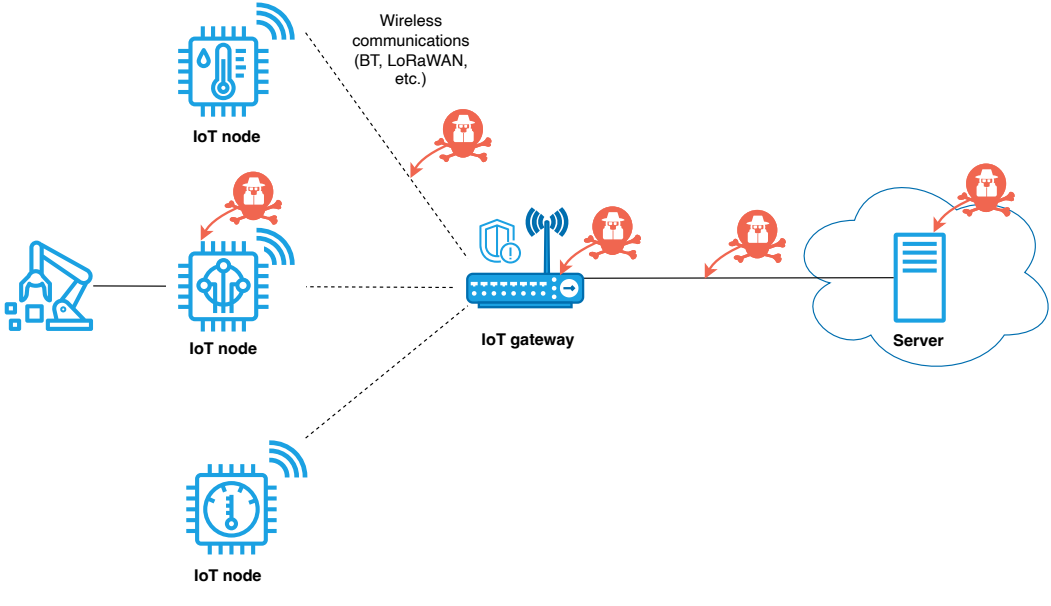
We must guarantee the best level of security for such systems

- Increasingly widespread in critical infrastructures
- They handle sensitive data
- They increase the global attack surface of information systems

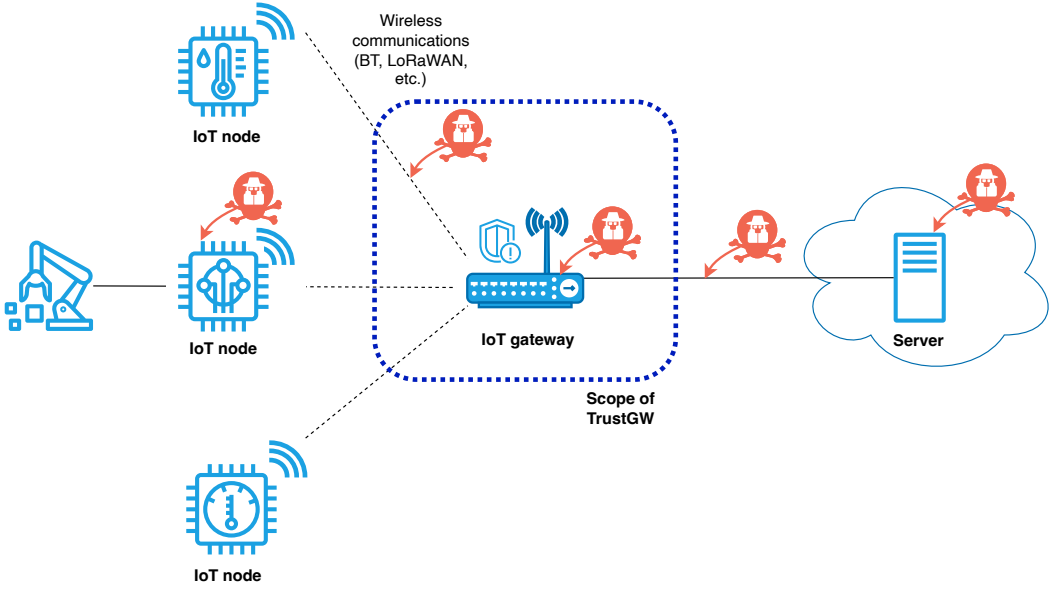
Threats against IoT architecture



Threats against IoT architecture



Threats against IoT architecture



TrustGW research project

Goal

Developing a dynamically reconfigurable and trusted heterogeneous software-hardware IoT gateway architecture

General information

- Started in January 2022. Duration: 44 months
- Funding (ANR): 3 PhD students + travels

Partners

- IRISA - CIDRE group (CentraleSupelec/Inria, Rennes)
- IETR - SYSCOM group (INSA, Rennes)
- Lab-STICC - ARCAD group (Univ. of South Brittany, Lorient and Brest)

Assumptions and scientific challenges

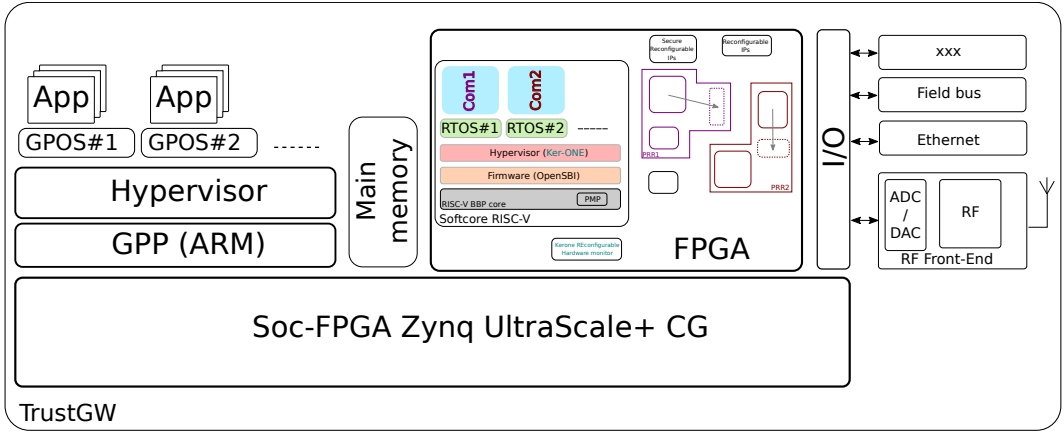
Assumptions

- The gateway is connected to IoT devices using different wireless technologies (BT, LoRaWAN)
- Different tenants share the gateway
- The gateway relies on a heterogeneous architecture (different CPU + hardware accelerators on FPGA)

Challenges

- Designing a trusted heterogeneous hardware/software architecture with dynamic reconfiguration capabilities to protect wireless communications
- Developing a trusted hypervisor to share all hardware resources (including FPGA) and isolate the different applications
- **Protecting edge-computing applications from software attacks**

General architecture



Use-cases and edge-computing applications

Device monitoring

- Monitoring events from smart sensors to detect malfunctions (e.g. predictive maintenance)
- FPGA-accelerated runtime verification of specifications written in TeSSLa

Machine Learning application

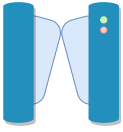
- Using ML to classify event traces
- Deploying the inferred model on FPGA to accelerate the classification

Cryptographic application

- Providing cryptographic primitives (signature, encryption) to protect application data
- Offloading part of the computation on FPGA

Preventive Approaches

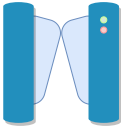
Preventing the malicious exploitation of a vulnerability



Access Control

Preventive Approaches

Preventing the malicious exploitation of a vulnerability

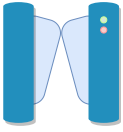


Access Control

Data encryption

Preventive Approaches

Preventing the malicious exploitation of a vulnerability



Access Control

Data encryption

Static analysis

Preventive Approaches

Preventing the malicious exploitation of a vulnerability

Preventive approaches are not **sufficient** to protect against all attacks

Access Control

Data encryption

Static analysis

Reactive Approaches

Monitoring the system at runtime to **detect** intrusions and **react**

Reactive Approaches

Monitoring the system at runtime to **detect** intrusions and **react**

Misuse-based detection using attack signatures

(model = malicious behavior)

\+ **Simple to enforce**

∼ **Few false positives**

– **Requires precise knowledge of the malicious code**

Reactive Approaches

Monitoring the system at runtime to **detect** intrusions and **react**

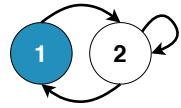
Misuse-based detection using attack signatures

(model = malicious behavior)

\+ **Simple to enforce**

∼ **Few false positives**

– **Requires precise knowledge of the malicious code**



Anomaly detection

(model = legitimate behavior)

\+ **The model does not depend on any information about attacks**

– **The model of legitimate behavior must be complete**

Reactive Approaches

Monitoring the system at runtime to **detect** intrusions and **react**

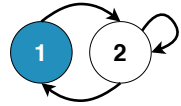
Misuse-based detection using attack signatures

(model = malicious behavior)

\+ **Simple to enforce**

∼ **Few false positives**

– **Requires precise knowledge of the malicious code**



Anomaly detection

(model = legitimate behavior)

\+ **The model does not depend on any information about attacks**

– **The model of legitimate behavior must be complete**

Reactive Approaches

Monitoring the system at runtime to **detect** intrusions and **react**

- How to define the reference behavior in order to avoid false positives?
- How to detect anomalies?

+ Simple to enforce

~ Few false positives

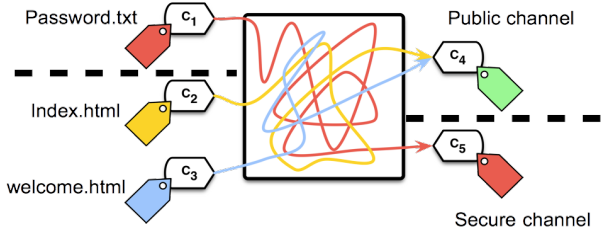
– Requires precise knowledge of the malicious code

information about attacks

– The model of legitimate behavior must be complete

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**



Different levels of DIFT

Coarse-grained approach: OS level

- Monitor system calls: containers = files, memory pages
- Pros & cons
  - \+ Monitor in kernel side protected from userland
  - \+ Tagging files is easier for the end user to specify its security policy
  - \+ Low runtime overhead
  - \- Over-approximation of application internal behavior
  - \- Cannot detect low-level attacks

Fine-grained approach: machine language level

- Monitor instruction execution: containers = registers, memory words
- Pros & cons
  - \+ Precise monitoring
  - \- No isolation if implemented in software
  - \- Cannot tag persistent storage (files) if implemented in hardware

Challenges

- Using **commodity Operating Systems**, able to execute **existing applications**

Challenges

- Using **commodity Operating Systems**, able to execute **existing applications**
- **ASIC** processor (i.e., without any modification)

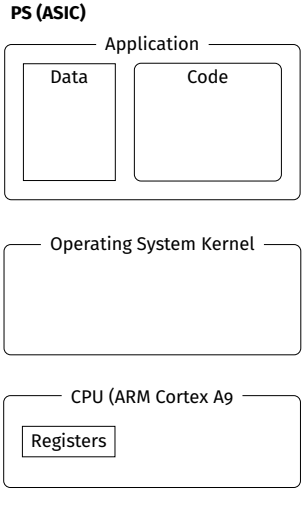
Challenges

- Using **commodity Operating Systems**, able to execute **existing applications**
- **ASIC** processor (i.e., without any modification)
- Handle **different types** of information flows
  - Registre $\longleftrightarrow$ Registre
  - Registre $\longleftrightarrow$ Memory
  - Memory $\longleftrightarrow$ Files

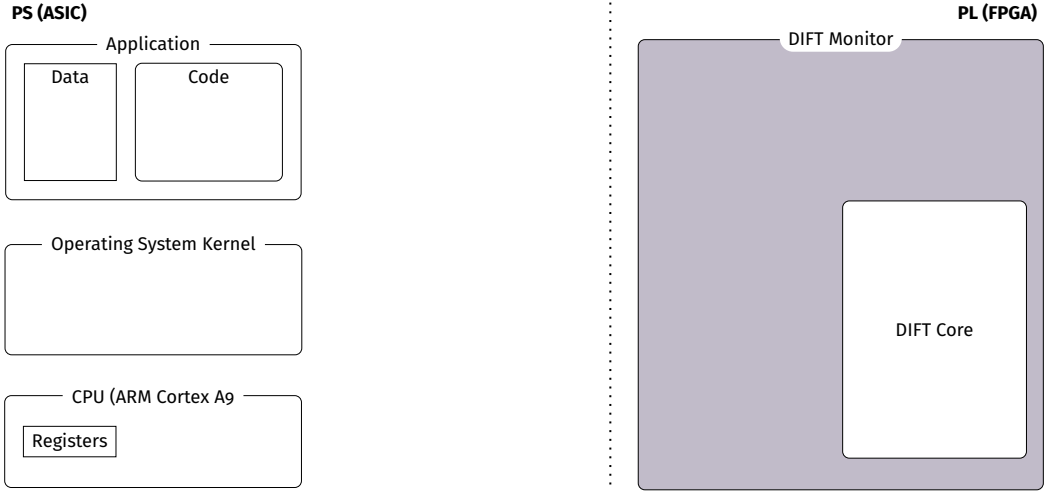
Challenges

- Using **commodity Operating Systems**, able to execute **existing applications**
- **ASIC** processor (i.e., without any modification)
- Handle **different types** of information flows
  - Registre $\longleftrightarrow$ Registre
  - Registre $\longleftrightarrow$ Memory
  - Memory $\longleftrightarrow$ Files
- Manage **tag persistency** for the filesystem

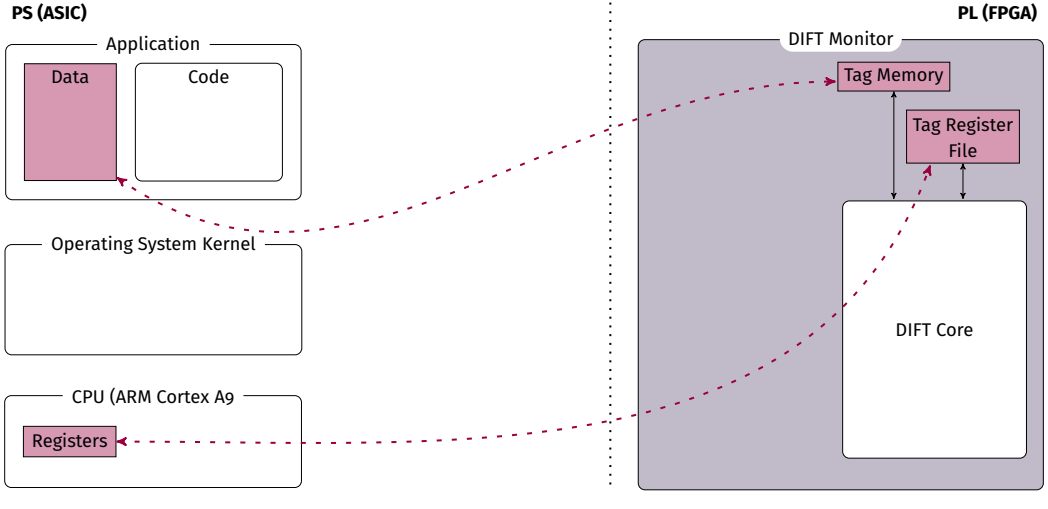
HardBlare : General Overview



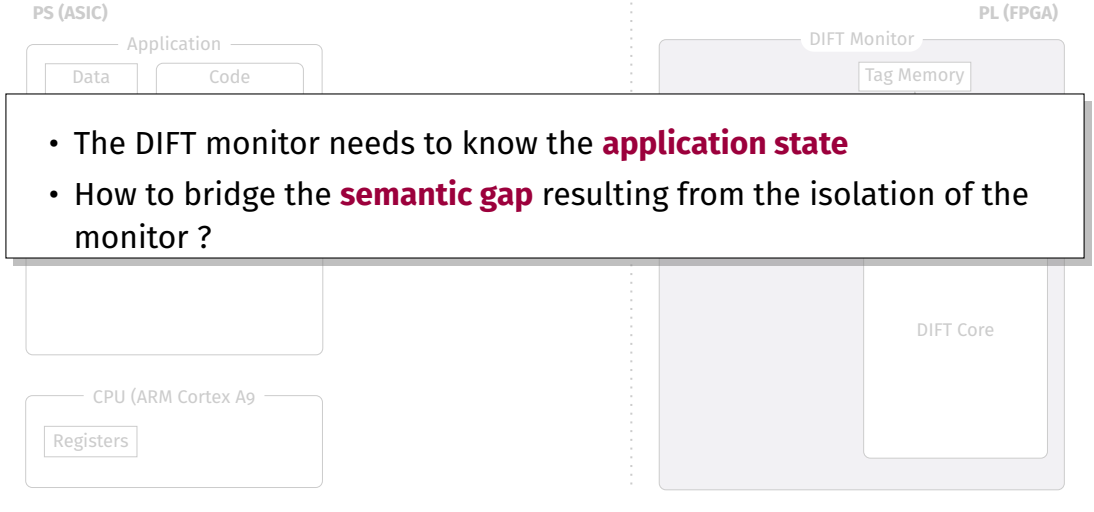
HardBlare : General Overview



HardBlare : General Overview

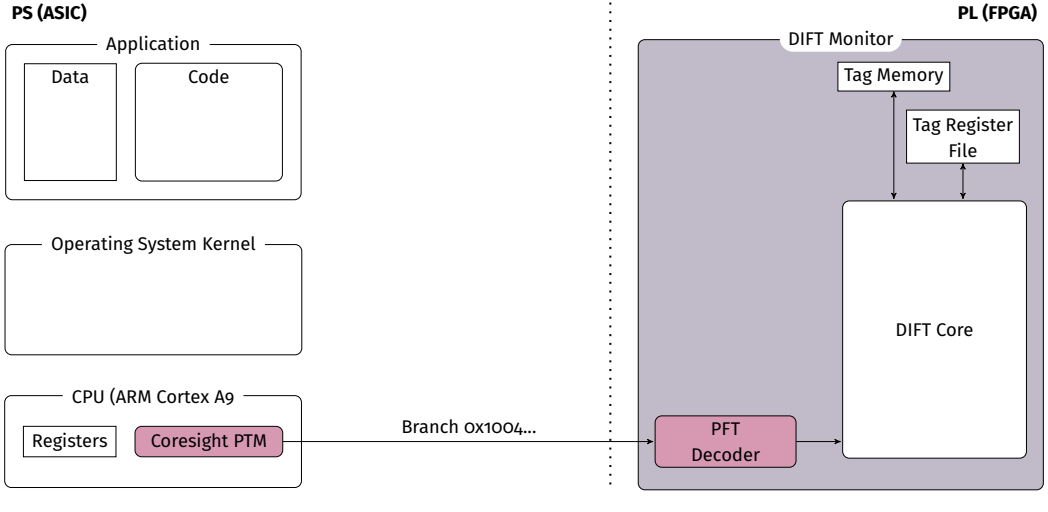


HardBlare : General Overview

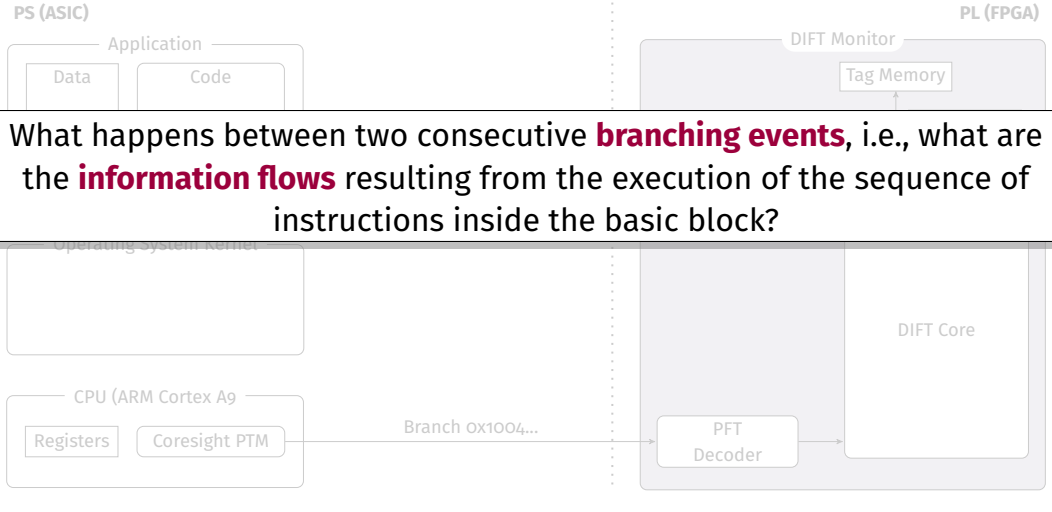


- The DIFT monitor needs to know the **application state**
- How to bridge the **semantic gap** resulting from the isolation of the monitor ?

HardBlare : General Overview

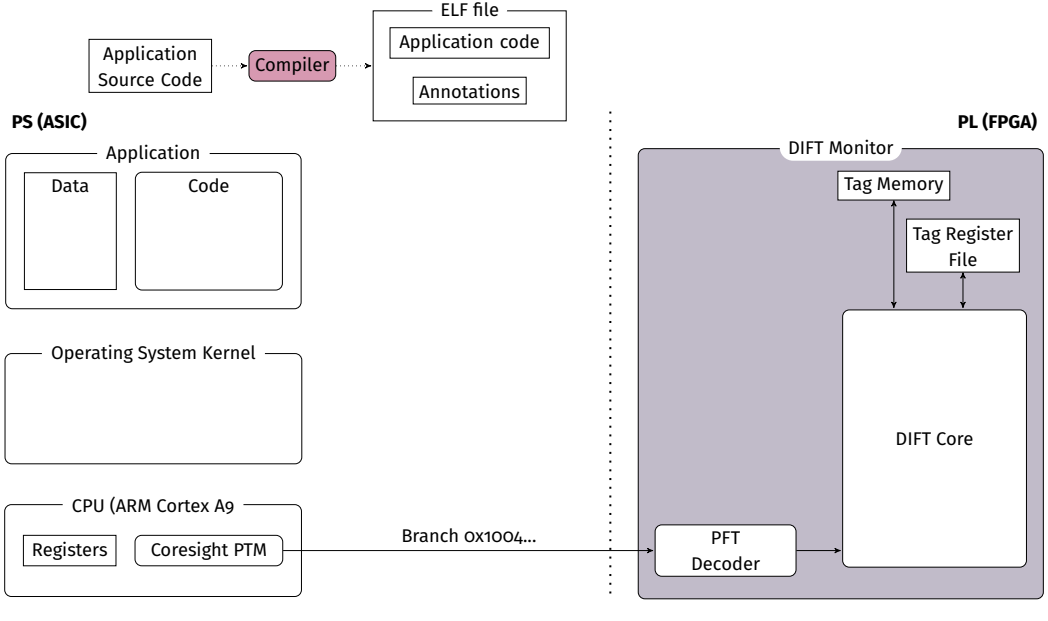


HardBlare : General Overview

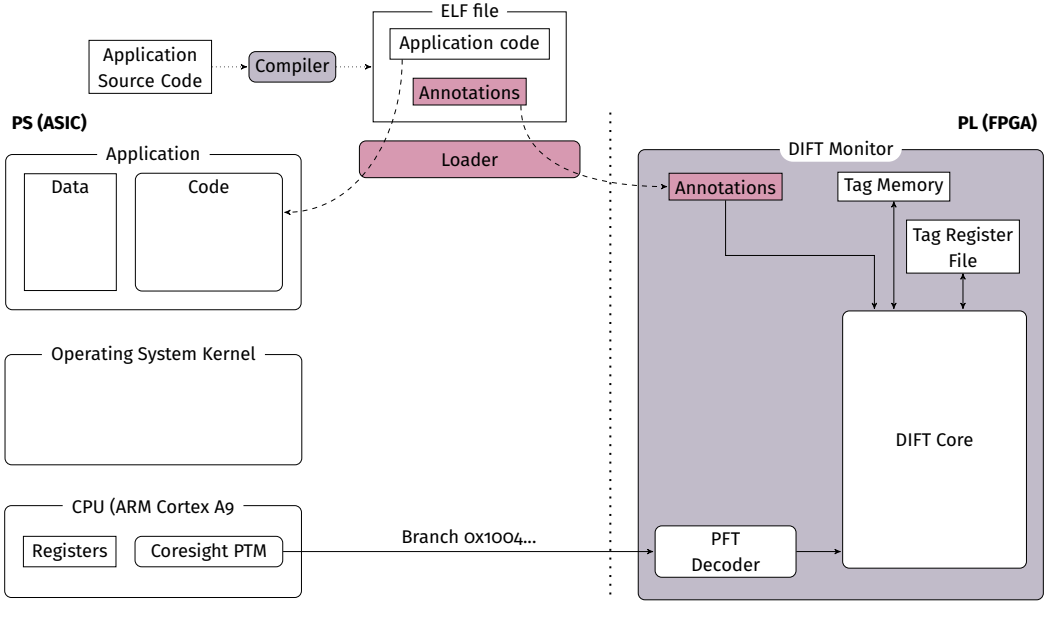


What happens between two consecutive **branching events**, i.e., what are the **information flows** resulting from the execution of the sequence of instructions inside the basic block?

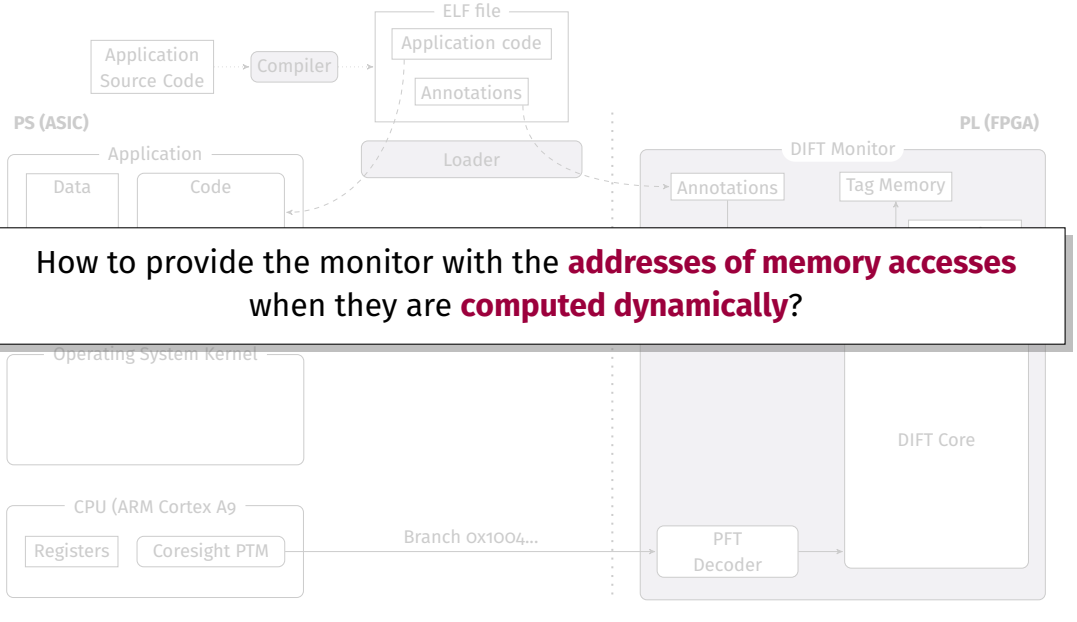
HardBlare : General Overview



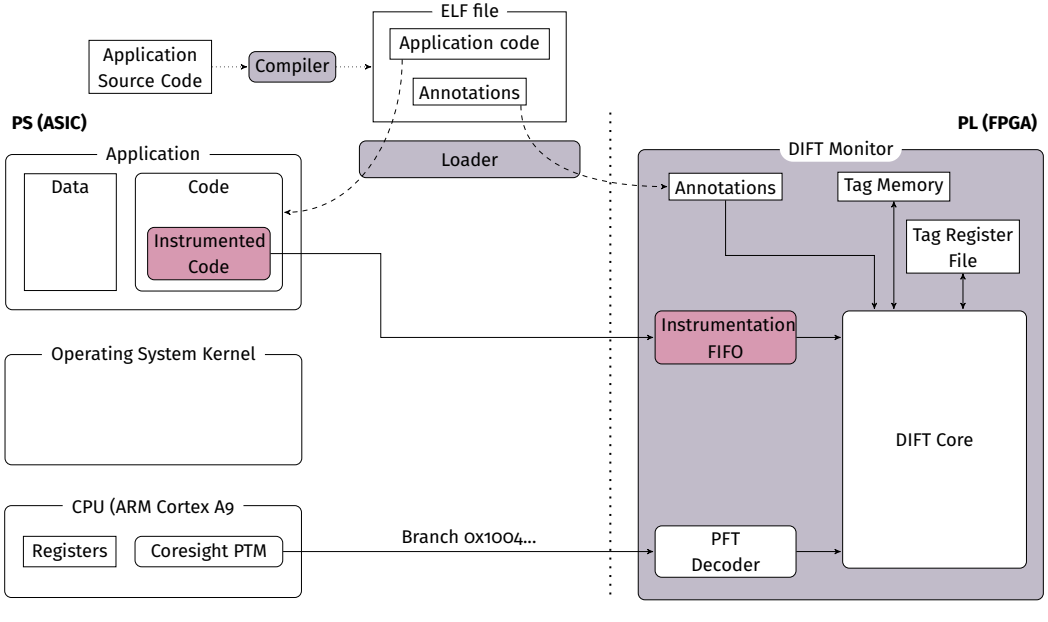
HardBlare : General Overview



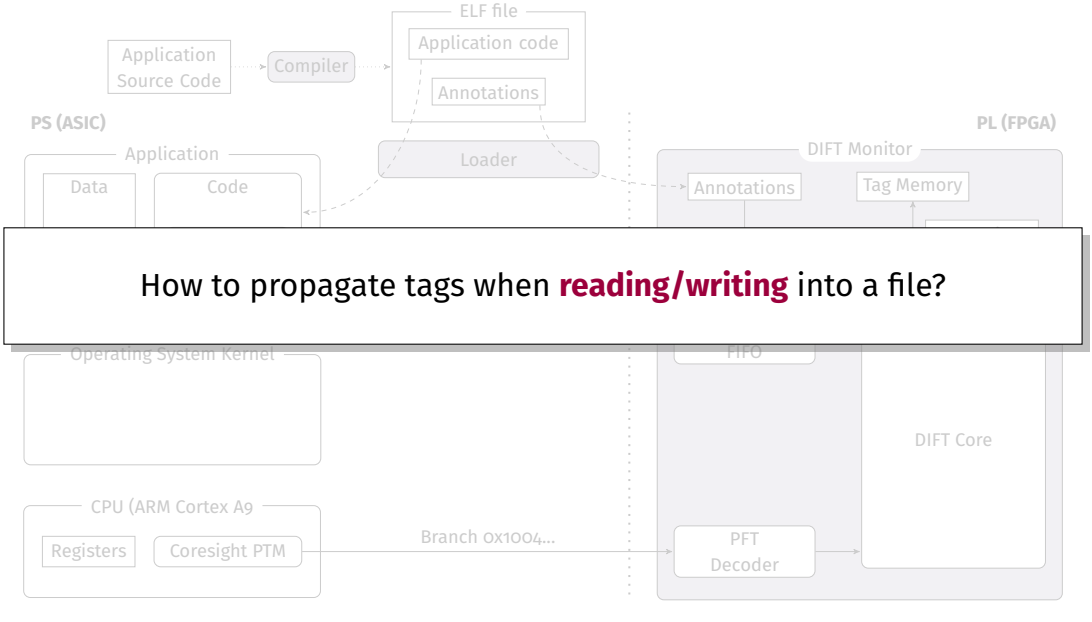
HardBlare : General Overview



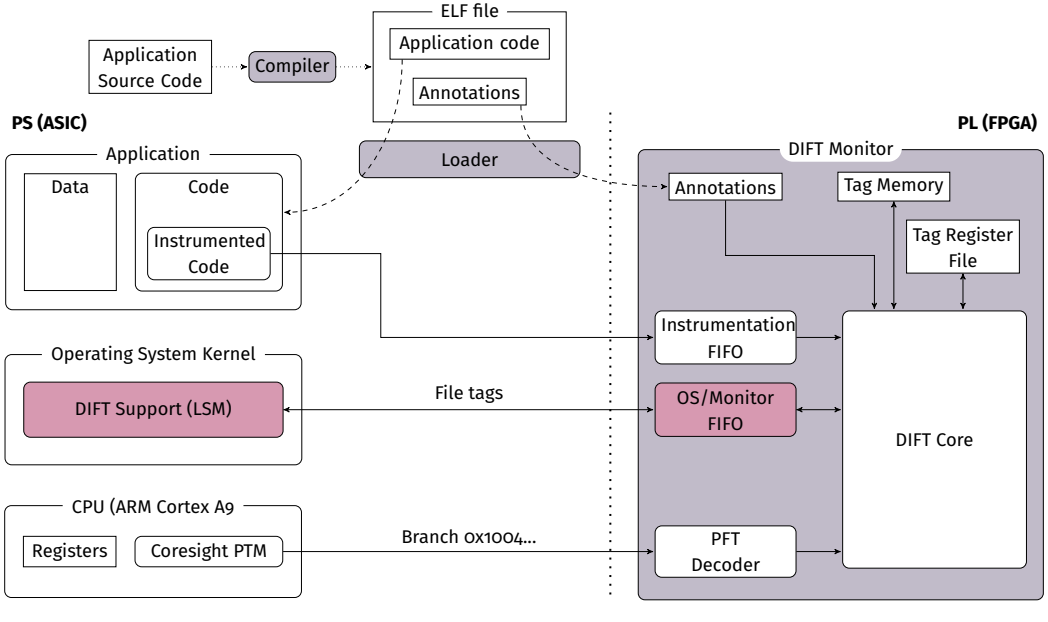
HardBlare : General Overview



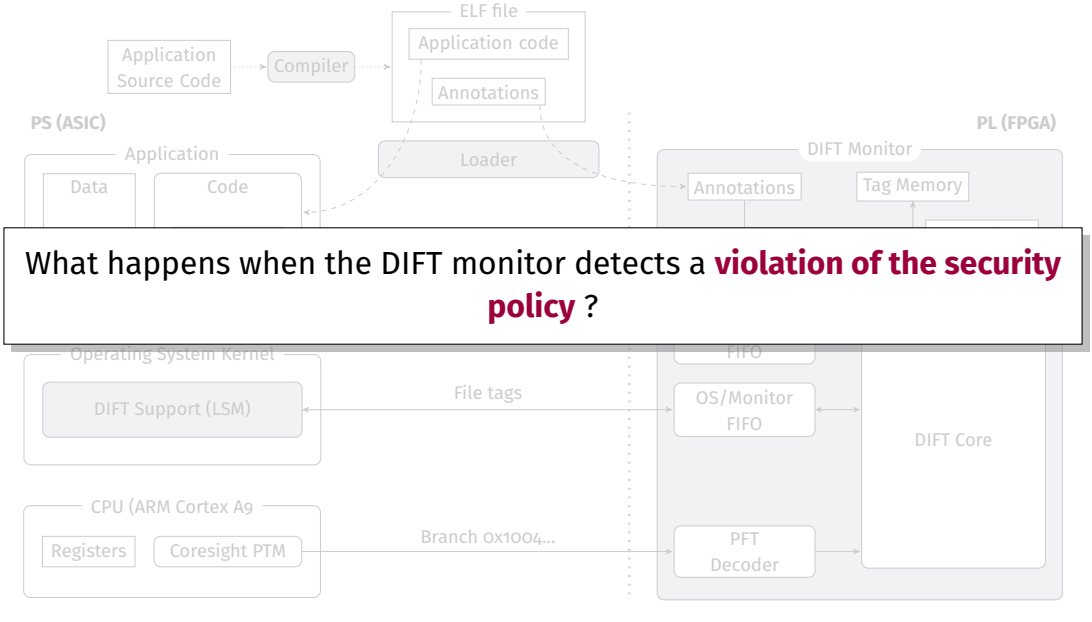
HardBlare : General Overview



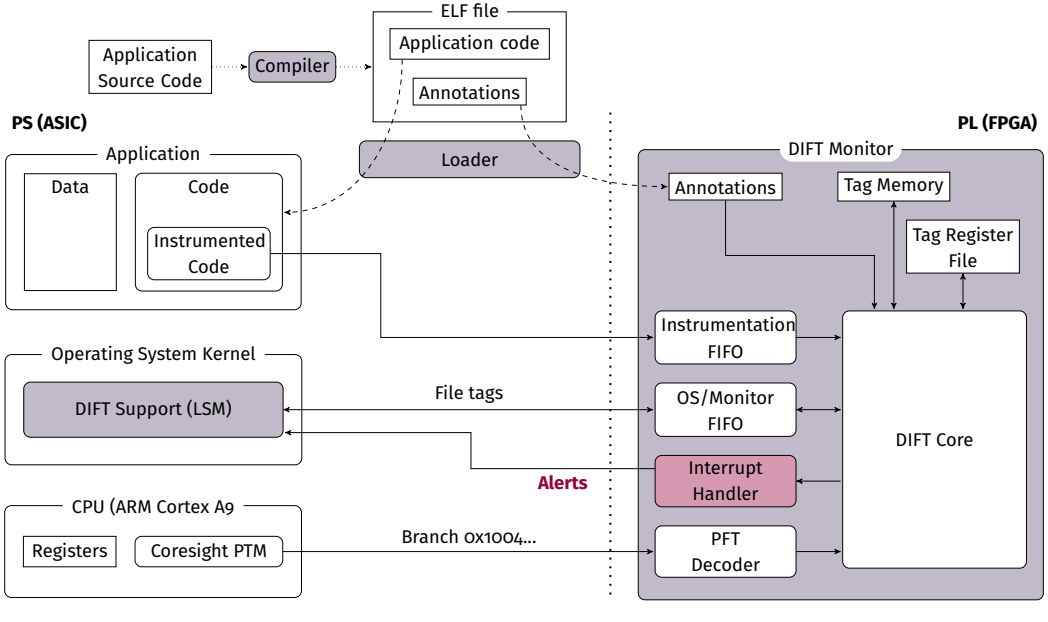
HardBlare : General Overview



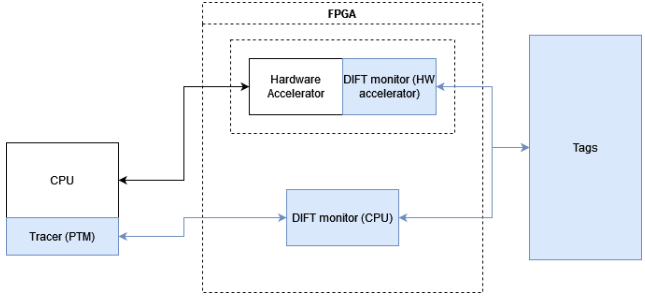
HardBlare : General Overview



HardBlare : General Overview

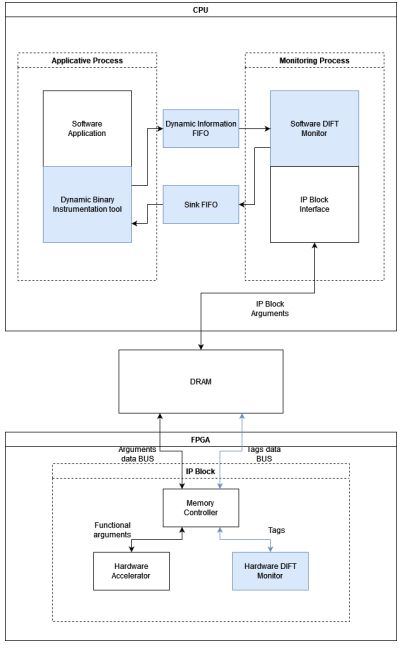


Monitoring Hybrid Applications : Ideal Design

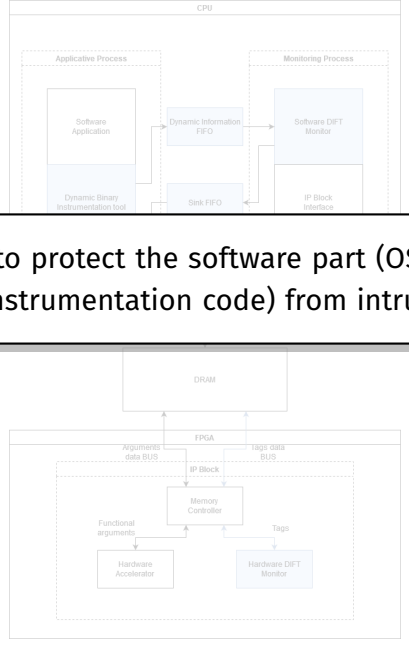


- Preliminary work realized by Romain Ninot during his master thesis
- We are recruiting (master-level internship, possibly PhD position)

A first software-based DIFT prototype

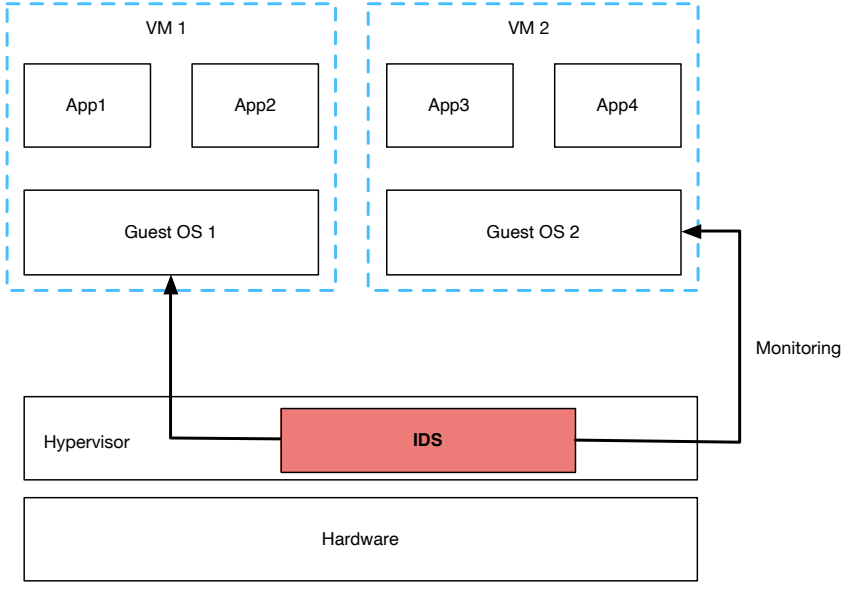


A first software-based DIFT prototype



How to protect the software part (OS kernel, instrumentation code) from intrusion?

Hypervisor-level IDS (PhD of Lionnel Hemmerlé)



Hypervisor-level IDS (PhD of Lionnel Hemmerlé)