COMIN Labs, focus Sécurité

Demande de paquet

HardBlare: Hardware and Software Information Flow Control

INRIA/SUPELEC CIDRE
guillaume.hiet@supelec.fr

Lab-STICC/UBS MOCS
guy.gogniat@univ-ubs.fr

IETR/SUPELEC SCEE
amor.nafkha@supelec.fr

January 5, 2014

Contents

1 Summary

The last decade of cyber defense has shown that for every security vulnerability discovered and patched, attackers find multiple alternatives to overcome these protection mechanisms [42]. One way to increase the security level of computer systems is to rely on both software and hardware mechanisms in order to allow a strong analysis of potential threats. In this context, the HardBlare project 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. Such an approach allows designer to build an efficient and reliable Trusted Computing Base. The general context of the HardBlare project is to address Dynamic Information Flow Tracking (DIFT) that generally consists in attaching marks to denote the type of information that are saved or generated within the system. These marks are then propagated when the system evolves and information flow tracking is performed in order to guarantee a safe execution and storage within the system. The HardBlare project builds on top of a standard software and hardware platform. The goal is to make no modification of the main processor core and to implement hardware DIFT in a dedicated coprocessor. Standard existing OS (like Linux) will be considered to execute existing applications. The HardBlare project relies on a combination of OS-level monitoring with a more precise language-level monitoring to produce a highly secure software hardware computing platform, featuring:

- A flexible approach where both software and hardware security rules can be updated dynamically in a per application basis or in order to update security rules when new threats are discovered;
- A large set of security rules with variable tag sizes and formats depending on the requirements;
- A persistent tag management feature;
- A cooperation mechanism between software static analysis and hardware DIFT to be able to monitor different types of information flow and so to address both confidentiality and integrity;
- A complete prototype of the system on a Xilinx Zynq board.

We believe the HardBlare project is ambitious and clearly goes beyond the state of the art. The HardBlare consortium is well balanced and complementary, all the partners have been active in research projects related to the main topics of the project. Furthermore, each partner has worked on different fields, thus bringing unique and complementary competencies to the consortium from software to hardware levels. Furthermore, no partner is unknown to all of the others. This will lead to a good collaboration among partners.

2 Description of the Project

Cyber security is now a major concern since a huge part of human activities depends on computers and networks. Users of such systems want strong guarantees on both confidentiality and integrity of their data. However, attackers tend to exploit vulnerabilities in computer systems to realize intrusions. Despite more than four decades of research in the field, such attacks are still possible. The motivations of the attackers have evolved and attacks tend to be more and more elaborated. For example, some malware are developed to take control of millions of machine which are connected to botnets once they have been infected. Those botnets are under the control of attackers that

can use compromised hosts to conduct massive attacks such as distributed denial of service, spam campaign or confidential data theft. This type of threat is strongly connected to the underground economy: malware, vulnerabilities or confidential information could be sold and botnets are rented as malicious services. Moreover, computer attacks are considered as a potential weapon that could be used in military interventions or by intelligence agencies. Personal computers are not the only targets since attacks could be conducted against critical systems such as Supervisory Control And Data Acquisition (SCADA) or Industrial Command Systems (ICS) in general.

On one hand system designers, developers and users in general are now aware of such security issues. They tend to adapt their behavior and to apply best practices. On the other hand, Information Technology systems have increasing complexity and keep changing, being more difficult to defend. A lot of security mechanisms are used to protect data confidentiality: access control and authentication, static analysis, testing, cryptography, etc. All these mechanisms are useful but, even when they are used in conjunction, they are not sufficient to prevent every intrusions. In this context, it is also necessary to monitor computer systems and to try to detect if such intrusions occur.

Many works have been done in the field of intrusion detection since the seminal work of Denning [12] and Anderson [1]. However, most of the Intrusion Detection Systems (IDS) that are used today rely on signatures of previously known attacks (and vulnerabilities). This approach is intrinsically limited since publicly unknown attacks, exploiting so-called zero-day vulnerabilities, could not be detected. Anomaly-based approaches, that only rely on a model of the legitimate behavior, seems appealing. In particular, in policy-based approaches, the user provides a specification of the security policy and the IDS monitors the system to check if this policy is satisfied.

Policy-based intrusion detection could be done using information flow control [51]. Indeed, both confidentiality and integrity violations are characterized by illegal information flows:

- in the first case, some confidential information leaks **to** an unauthorized (non-trusted) user or application;
- in the last case, some information is modified by an information flow **from** an unauthorized (non-trusted) user or application.

Policy-based approaches raise two scientific issues:

- How to specify a security policy that best describes the security needs of users?
- How to enforce or verify such a policy?

The first problem has been addressed by several works that proposed formal security policy models [3, 5, 8, 4, 39] or approaches that help users to specify their security needs [24, 2]. In particular, the CominLabs POSEIDON [1] project focuses on formal specification of security policies and automatic deployment of such policies. The POSEIDON project also studies the security guarantees resulting from the combination of different security mechanisms.

In this project, we focus on the second problem and more precisely on one specific mechanism: Dynamic Information Flow Control. We want to address three major issues related to this approach:

- How to decrease the overhead induced by the monitoring process?

[1] http://www.cominlabs.ueb.eu/themes/project/

- How to design a flexible solution that can address a wide range of threats while minimizing hardware and software modifications?
- How to protect the DIFC itself?

Such issues have not completely been addressed by previous works and represent strong limitations to the adoption of such mechanisms in real life systems.

2.1 Related Work

Information flow control (IFC) is a security mechanism that provides security guarantees about information propagation. Other security mechanisms such as access control or cryptography can be used to limit the dissemination of confidential information and the modification of high integrity contents. However, they do not enforce end-to-end properties: They cannot control the dissemination of information once file access is allowed or the data is decrypted.

Following the seminal work of Denning [13], the majority of the efforts in this domain have focused on language-based static approaches [40]. Most of them rely on security type systems following Volpano and Smith approach [45]. They consist in attaching a mark, called security label or tag, to some information containers, such as program variables, or input interfaces. This mark denotes the type of information that is saved or generated by such "information sources". Security labels are generally organized in a security lattice that models the information flow policy [11, 32]. The static analysis verifies that some form of non-interference [17] is satisfied between variables associated to incompatible labels (typically, between secret and public variables). Such approaches have been proposed to control information flows in Java [31] and OCaml [36]. However, despite more than three decades of research, we must admit that this type of approaches have not been widely adopted in systems used in real life [48]. Several limitations of such static analysis could explain this situation:

- In such approaches, security labels are bind to source code at verification time. Thus it is difficult to handle variables than can be used to save information of different sensitivity. In practice, the security labels of many program inputs depend on the data accessed and could only be known at execution time.
- These approaches are not suited to verify a whole system including many software pieces written in different languages. In particular, it is difficult to control information flows between different applications.
- These approaches are very restrictive as they reject programs if only one execution could lead to illegal information flows. Moreover, static analysis is sound by design but they could reject programs that satisfies the security policy.

These limitations motivate the use of Dynamic Information Flow Control (DIFC) [41]. In such approaches, security labels are propagated to information containers when the system evolves and information flows occur. The verification of the policy is performed when some information containers are modified or when some data are sent to output interfaces. Two main characteristics can differentiate DIFC approaches that have been proposed in previous works:

1. The level of monitoring and consequently the granularity of DIFC;
2. The types of information flows that are taken into account.

2.1.1 Different levels of monitoring

DIFC can be achieved at the operating system level. This coarser-grained monitoring technique attaches labels to information containers such as files, memory pages or IPC. Some works have led to dedicated operating systems such as Histar [49] or Asbetos [14]. These approaches rely on new design choices that facilitate DIFC. However, they are incompatible with existing applications and hardware drivers, which limits there adoption.

Existing OS security modules such as SELinux [29] or TrustedBSD [46] implement Mandatory Access Control (MAC). To some extent, they can be used to limit information flows in the system. However, the lack of flexibility of MAC limits its usage. On the other hand, approaches that rely on DIFC offer more flexibility. Flume [27] is one of the more recent attempt to implement DIFC on standard OS (Linux). Therefore, the monitor is implemented in user space and cannot rely on kernel space isolation protection. The SUPELEC CIDRE team also developed Blare[2], a security module for the existing Linux kernel [15, 21]. Blare monitor resides in kernel space, which offers better protection. This OS-level DIFC is also fully compatible with existing applications and drivers. No modification is required on the source nor on the binary code of these software pieces.

OS-level DIFC is interesting for several reasons:

- This approach is well suited for inter-process information flow monitoring as the OS is the central point on a computer;
- OS-level approaches can easily attach persistent labels to files and enforce DIFC across reboot;
- OS kernel is isolated from user space applications thanks to hardware mechanisms, thus preventing malicious or compromised applications to interfere with the monitoring process;
- The overhead of the monitoring is limited (about 10%).

However, all theses approaches suffer from the over-approximation problem that could lead to the label creep issue [40]. Indeed, processes are seen as black-boxes and the monitoring system considers that every write access depends on all the read accesses that were performed previously by the same process. This over-approximation of the actual behavior of the process could lead to false positives.

One solution to tackle this problem is to rely on a more precise language-level monitoring to track information flows inside applications. In this case, labels are attached to fine-grained containers such as variables or memory bytes. The DIFC monitor is generally implemented within each application using weaving (AOP) [30], compilation [47, 28] or instrumentation [34, 37]. These approaches rely on binary code [34, 37], bytecode [19] or source code [47, 20, 28] modifications. Another solution consists in modifying the runtime environment, for example the virtual machine or the interpreter used in languages such as Java [7, 33] or PHP [35]. The SUPELEC CIDRE team also developed JBlare [3], a language-level DIFC monitor that tracks information flows inside Java applications [23, 22]. More recently, the CominLabs SecCloud project aims at using DIFC to control information flow inside JavaScript, at the browser level. The HardBlare project is complementary to the SecCloud project as we want to offer hardware facilities to enhance DIFC. Moreover, SecCloud only target specific attacks relevant to JavaScript and browser environment.

[2] https://blare-ids.org
[3] https://blare-ids.org/flavors/jblare/

2.1.2 Monitoring different types of information flows

Different types of information flows could be taken into account depending of the DIFC approach. All the works presented here do not take into account side channels such as timing channels. This type of information flows could be used by an attacker to infer confidential information by monitoring execution time or power consumption. However, such attacks suppose to have physical access to a protected hardware (smart cards, crypto-processors, etc.). In the HardBlare project, we do not consider attacks targeting such protected environment.

Most DIFC approaches [34, 37, 19, 47, 28, 35] rely on tainting techniques and only consider direct flows resulting from affectations and computations. Such approaches are sufficient to track information flows that could lead to integrity violation. Thus tainting is often used in conjunction with sanitization functions. These approaches generally consist in verifying that information coming from untrusted sources are not used directly to manipulate sensible data but are first passed to sanitization functions.

Few works [7, 33] try also to take into account indirect flows resulting from conditional branching. Taking into account such indirect flows is necessary to detect all confidential data leakage. Indeed, if an attacker could inject some code, he can craft a malicious code that leaks confidential data using indirect flows. However sound detection of indirect flows is difficult because it supposes to take into account affectations in non executed branches of a given program run. This supposes to rely on hybrid approaches [7, 33, 18] that combine DIFC with static analysis.

Language-level software-based DIFC is appealing but the gain of precision comes at the price of a huge overhead [10] (from 3 x to 37 x), which limits the usefulness of the solution to detect attacks. Indeed, contrary to vulnerability analysis, using DIFC as a policy-based IDS supposes to monitor applications when they are executed by users. In this case, the DIFC overhead is a major concern. Moreover, in such approaches, the monitoring mechanism is not isolated anymore from malicious code. To be trusted, it needs to implement some form of auto protection.

2.1.3 Combining different monitoring levels

As explained in previous sections, OS-level DIFC is interesting as it can easily monitor inter-process information flows with little overhead and support tag persistency. One solution to tackle the over-approximation issue of such approach is to combine OS-level with language-level monitoring. Thus, OS-level DIFC tracks inter-application information flows and manages labels of coarse grained persistent containers (e.g., files). It can be configured by the administrator to manage the policy by attaching tags to persistent containers. Language level DIFC tracks intra-application information flows and cooperates with OS-level to inherit tags from OS-level (read accesses) or propagate tags to OS-level containers (write accesses). Laminar [38] in one of the few works that have proposed such cooperation. However, application should be modified by their developers to take advantage of Laminar language-based DIFC. JBlare, developed by the SUPELEC CIDRE team, also relies on OS-level and language-level DIFC cooperation [22] but does not require any modification of the source code of monitored applications.

This approach is appealing but two issues remain:

- How to significantly decrease the overhead of language-level DIFC?
- How to protect language-level DIFC?

2.1.4 Hardware assisted DIFC

The HardBlare project tackles these last issues by combining software and hardware mechanisms in order to build an efficient and reliable monitoring of information flows. Hardware DIFT consists in modifying existing hardware to accelerate tags propagation and computation. Hardware mechanisms could also be used to protect tags. A cooperation with software-level (OS and/or applications) is necessary to manage the tags and the policy.

There have been several efforts in the domain of hardware DIFT. One of the first contributions was provided in 2004 by Suh et al. [43]. In their work they have developed a complete solution in order to demonstrate that extending processor core was an efficient solution to improve DIFT efficiency and to reduce performance overhead. Their work was validated through simulation and a reduced set of security policies was considered.

Based on this first proposition several efforts have been conducted, the work of Dalton et al. in 2007 [10] with the Raksha contribution and more recently the work of Chiricescu et al. [42] with the SAFE solution can be mentioned. These approaches are very interesting and provide actual implementations to demonstrate their contribution about hardware extension of processors and OS modification to build an efficient solution. The main drawback of these solutions is that they do not rely on generic processor and OS but require the development of some dedicated solutions which can limit their adoption. Validating a new processor and OS is a very long process and in practice processor designers are generally not ready to pay the price.

In 2009, Kannan et al. [25] have addressed this issue and have proposed a dedicated hardware coprocessor to perform tag analysis. Such an approach is interesting as it demonstrates that a less intrusive solution is possible and efficient. The synchronization between the application running on the processor and the tag analysis computed in the coprocessor is performed at each system call. A specific interface between the processor and the coprocessor allows for an efficient communication between both components (but still at the price of few modifications of the processor core to retrieve some specific information). The considered tags are 4-bit long and high to low levels verifications are performed. An actual prototype of the system has been developed.

In 2008, another approach has been proposed by Venkataramani with the FlexiTaint solution [44]. Their solution is very interesting as it proposes not a single security policy but a flexible approach where the security rules can be adapted for each application. They also provide some efficient caching approaches in order to reduce the overhead when performing the tainting of data. This solution is the most advanced one concerning flexibility of security policies but still needs some modifications of the processor architecture to implement their contribution.

A last interesting work has been proposed by Hari Kannan in 2009 [26]. The author proposes a comprehensive hardware solution to address the correctness and performance challenges of decoupling application execution and dynamic analysis in multiprocessor systems. His scheme ensures consistency between application data and analysis metadata. This approach is based on two processor cores: an application core and a separate analysis core. Such an approach is interesting but requires a specific processor to perform DIFC. Many applications cannot support such a high area overhead.

2.2 Scientific contributions of the project

The main objective of this project is to design efficient hardware support to reduce DIFC overhead. We believe that this effort will promote the adoption of such security mechanisms. We want to

propose solutions that could be implemented by industrial partners in medium-term. In this context, compatibility with existing software/hardware is mandatory. We must also satisfies the following constraints to justify the cost of the hardware modification:

- the solution must be flexible (i.e. policy agnostic and tag format agnostic) so that it could be used in different contexts;
- the solution must be able to track both direct and indirect flows to detect a wide range of attacks (both integrity and confidentiality attacks);
- the overhead must be acceptable to users that are looking for security hardened environment (for example, governmental or military usage).

None of the previous works satisfies all these constraints, which explains that DIFC has not been widely adopted to protect computer systems contrary to other security mechanisms such as access control or cryptography. OS-level DIFC approaches suffer from the over-approximation issue that limits there ability to detect a wide range of attacks. Language-level software-based DIFC exhibits huge overhead. Modifications of the processor core are the main limitations of existing hardware DIFC approaches. Even the less invasive solutions [25, 44] rely on modifications of the main core. This limits the adoption of such approaches as implementing them implies to convince semiconductor chip makers to modify the architecture of their microprocessor or to implement the modified core on FPGA. This last solution is limited as only simple processors could be synthesized and the performance of such FPGA-based CPU cannot compete modern CPU. Moreover, all existing hardware DIFC approaches only track direct flows, which limits their ability to detect information leaks.

Our project aims to go beyond existing solutions by providing the three following scientific contributions:

- Provide a solution based on a standard software and hardware platform.
- Provide a flexible approach where both software and hardware security rules can be updated dynamically in a per application basis or in order to update security rules when new threats are discovered.
- Provide a cooperation mechanism between software static analysis and hardware DIFT to be able to monitor different types of information flow and so to address both confidentiality and integrity.

Regarding the first contribution, we want to make no modification of the main processor core and to only implement hardware DIFT in dedicated coprocessor using FPGA. This point is clearly a strong challenge as existing processor cores have not been built taking into account DIFC. However, there are solutions to retrieve required information to perform DIFC through processor I/O ports accesses and through monitoring L1 and L2 cache accesses. We believe using these approaches will allow us to propose a solution that do not require processor core modification. We also want to rely on standard existing OS (like Linux) and to be able to execute existing applications. In order to achieve this goal the HardBlare consortium will rely on his expertise in the domain of hardware design (processor architectures, embedded system design) and in the domain of software development (OS and language DIFC). On the software side this contribution will rely on the Blare operating system developed by SUPELEC CIDRE team that has already proven its efficiency to

monitor information flows. On the hardware side the UBS MOCS and SUPELEC SCEE teams have a strong background in developing software-hardware systems, especially on FPGAs. Based on these previous contributions we will be able to develop a complete software-hardware solution relying on standard technological components.

Regarding the second contribution, the HardBlare consortium has a unique expertise in the domain of dynamic reconfiguration for FPGAs through the UBS MOCS and SUPELEC SCEE teams. Many projects and demonstrators have been developed within these teams where such a technique was implemented. Some projects were not targeting security but others do (for example to protect internal and external communications within a multiprocessor system on chip where hardware security rules where modified at run-time). On the software side the expertise of the SUPELEC CIDRE team in DIFC will allow the proposition of a flexible solution where security policies will be dynamically updated.

Regarding the third contribution, the combination of hardware monitoring (at the coprocessor level) and software monitoring (at the operating system level) is a key point to build an efficient solution. The expertise of the HardBlare consortium in developing software and hardware systems will allow providing an effective solution. The SUPELEC CIDRE team has a strong background at the software level and UBS MOCS and SUPELEC SCEE teams have a strong expertise at the hardware and system levels.

Our project will also target the two following technical points in order to develop a complete demonstrator:

- Provide persistent tag management feature.
- Provide a complete prototype of the system on a Xilinx Zynq board.

Regarding persistent tag management feature, the HardBlare consortium will rely on the SUPELEC CIDRE team expertise. This team has already developed different solutions at the operating system level (Blare solution) in order to perform tainting of mass storage. Their approach will be extended to take into account the co-designed software-hardware approach in order to increase the efficiency of the current solutions.

Finally, the UBS MOCS and SUPELEC SCEE teams have already an expertise with the Xilinx Zynq board through different projects. This expertise will be a major advantage to develop a complete system where operating system and applications running on an ARM Cortex-A9 will cooperate with hardware monitoring techniques running on the coprocessor (FPGA part of the board).

We believe such a proposition is ambitious and clearly goes beyond the state of the art as illustrated in Table 1. We also believe the HardBlare consortium has all the skills required to address such a project and their unique complementarily will allow achieving a step forward existing solutions.

2.3 Project organization

The HardBlare project is composed of five work packages (WP) that allow to build a complete solution. WP1 leads to setup the foundation for the target software hardware system. WP2, WP3 and WP4 will developed the technological components required to build our proposition. WP5 will provide a full integration of the system. Figure 1 shows the connections between the different work packages. Each work package provides inputs to other work packages. This point is original as it will conduct to a full software-hardware integration.

Table 1: Scientific contributions of the project

State of the art	**Progress beyond the state of the art**
DIFT techniques with processor architecture modification	DIFT techniques without modification of the core processor architecture
Persistence not considered for software-hardware architectures	OS-level support and hardware/software cooperation to manage persistence
Hardware support for dynamic analysis of information flows to address integrity	Combined software-hardware support for static and dynamic analysis of information flows to address both integrity and confidentiality
Software flexibility	Hardware flexibility through dynamic reconfiguration

WP1 – Baseline solution for DIFT in hardware

In this work package the goal is to set-up all fundamental functionalities in order to develop an extended solution for reconfigurable DIFT in hardware. The targeted platform is a Xilinx Zynq board integrating two ARM Cortex-A9 processors. This board corresponds to an interesting target as it embeds several hard-IP processors and a large programmable logic area where specific functionalities (e.g. coprocessors) can be developed and connected to the processors. Such a board provides a large flexibility and allows for an efficient exploration of dynamic reconfigurable software-hardware systems.

In this work package the whole software stack will be developed and also a minimal hardware coprocessor implementing a reduced set of security policies. We envision connecting the coprocessor directly to the ARM Cortex-A9 processor through the accelerator coherency port (ACP) in order to have a direct access to the caches and the memory subsystem exactly the way the APU processors do. In this way the interaction between the software running on the processor and the hardware executed on the coprocessor (implemented in the programmable logic area) will be very tight. Such an approach will guarantee a high performance and efficiency of the solution we want to develop.

This work package will also analyze our architectural approach compared to the state of the art solutions when a coprocessor is considered. Compared to existing efforts our solution does not require any modification at the AMR Cortex-A9 level, which is a key point in order to have a large adoption of such a solution. This also implies technical challenges that must be solved to design an effective and efficient solution.

WP2 – Combining static analyzes with DIFT

In this work package the goal is to develop original solutions in order to monitor both direct and indirect (depending of conditional execution flow) information flows. This implies to be able to

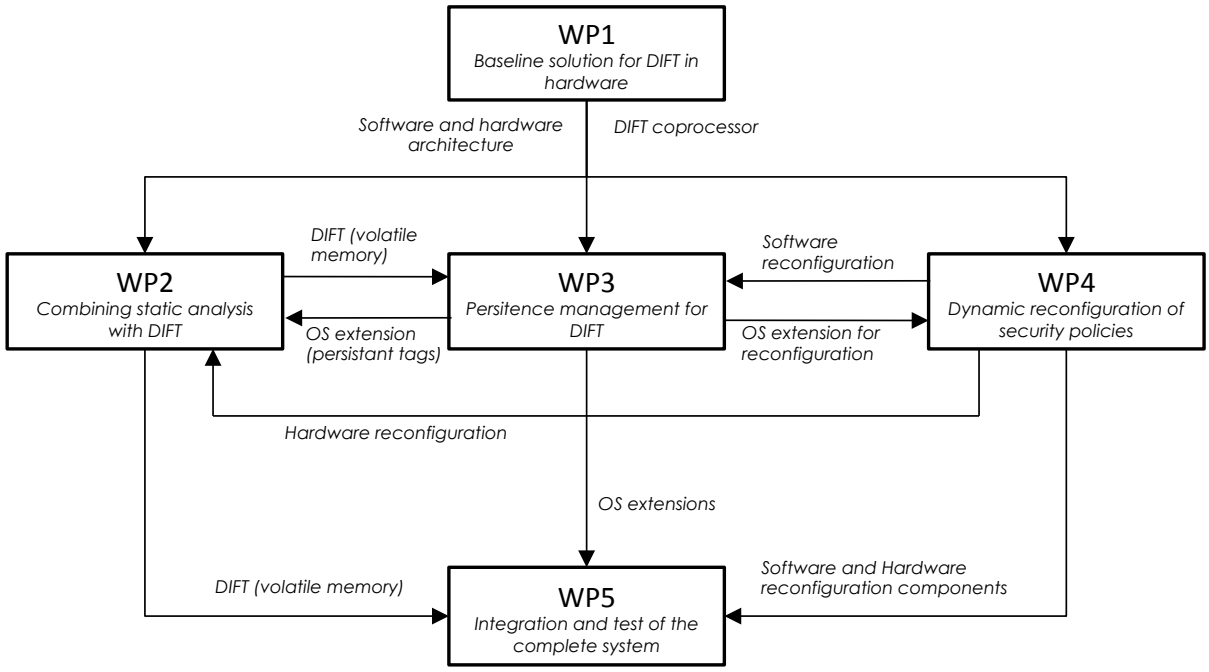

Figure 1: Organization of the project

combine DIFT, accelerated by hardware coprocessor, with software static analyzes. The later should be performed before dynamic monitoring (off-line or at loading time). It will help the dynamic monitoring to handle indirect flows by adding annotations to the machine code (for example, using dedicated instructions). Current solutions do only target dynamic tainting and so can only handle direct information flows. Our approach goes a step forward in the way security is maintained during the execution flow of the application. The solution we propose will be able to detect a wider range of attacks, including data leakage. In order to provide an efficient and a reliable solution we will base our approach on contributions developed within the SUPELEC CIDRE research group to combine static and dynamic approaches at software level.

WP3 – Persistence management for DIFT

In this work package the goal is to provide tainting for mass storage, i.e. files, in order to provide confidentiality and integrity protections for the whole system. This persistent tag management is necessary to maintain the security level when the system is rebooted. It also helps to monitor information flows across different applications, i.e. when different applications use files and IPC to exchange information. Such an approach relies on the definition of specific services at the OS level. In our case we promote a solution based on existing and widespread OS, so the goal is not to develop a dedicated OS but to leverage existing ones.

We will base our approach on contributions developed within the SUPELEC CIDRE research group to monitor information flows at OS-level (kBlare). We plan to modify the existing prototype so that it can take advantage of the DIFT coprocessor. This implies to define and to develop a cooperation mechanism between hardware-level and OS-level DIFT.

WP4 – Dynamic reconfiguration of security policies

In this work package the goal is to propose a full flexible solution both at the software and at the hardware levels. It is essential to provide developers with security policies that can be updated for each new application and when new threats are developed in order to provide specific and up-to-date security protections. At the hardware level we can take benefit of the reconfigurable platform in order to dynamically change the architecture and provide an efficient solution for each application. The frequency of the reconfiguration rate can be variable as it can be on an application execution basis or on an updating basis. In order to provide a trusted solution, OS services and hardware mechanisms will have to be developed. Expertise of all partners will be a key point in order to develop an efficient solution.

WP5 – Integration of the complete system for an extended DIFT software and hardware approach

In this work package the goal is to gather all developed contributions and to perform extensive test to demonstrate the effectiveness and the performances of the system. This work package aims to demonstrate that DIFT techniques can be performed in an efficient way without developing dedicated processor cores and OS. We plan to conduct two types of experiments. First we want to measure the overhead of the global DIFT mechanism by executing typical use-case scenario and performance test suites. We want also to verify that we can detect a large range of attacks with few false positives. Figure 2 shows a simplified view of the final architecture and associated services.

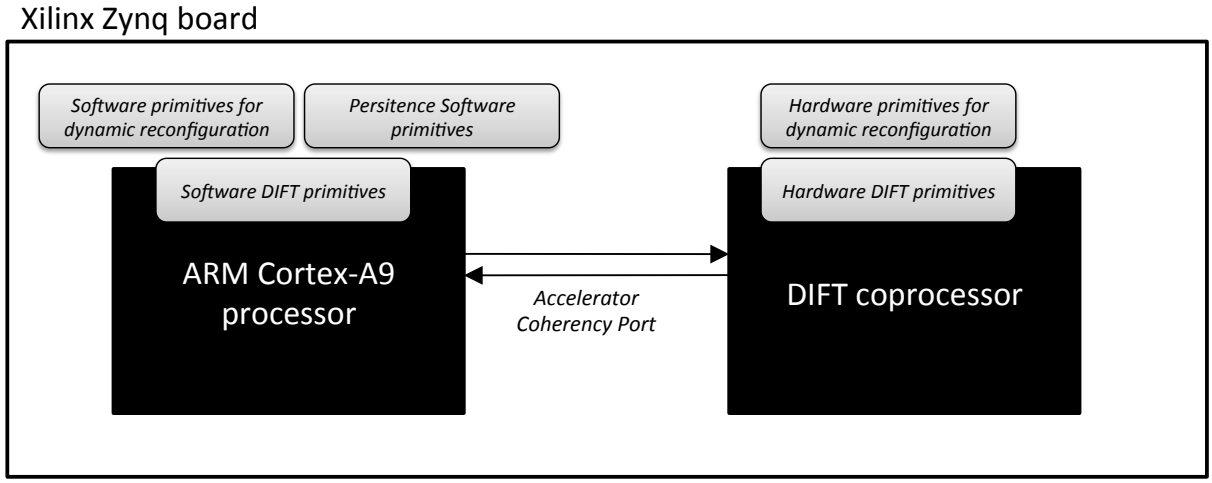

Figure 2: Hardware architecture

2.4 Key expertise of the consortium to address the HardBlare project

The composition of the HardBlare consortium is well balanced and complementary, all the partners have been active in research projects related to the main topics of the project. One key point regarding the consortium is the multidisciplinary skills brought by the three teams, spanning from

software to hardware and from software security to hardware security. This expertise is mandatory to address the HardBlare project.

SUPELEC CIDRE has been working for years on software security, especially on Dynamic Information Flow Tracking [3, 50, 51, 22, 23]. We developed DIFC prototypes [4] at OS-level (kBlare) and language-level (jBlare). Latest version of JBlare rely on hybrid static/dynamic analysis. We also developed cooperation mechanism between OS and language level monitoring. SUPELEC CIDRE will bring its expertise in software development and Linux kernel modifications.

SUPELEC SCEE has been working for years on dynamic and partial reconfiguration for FPGAs, processor architectures and embedded systems design [5].

UBS MOCS has been active for more than ten years in the field of embedded systems security and, in particular, monitoring techniques for embedded systems [6, 9, 16]. Several projects have been performed these last years dealing with embedded system security and reconfigurable technologies. The SecReSoC project [6] goal was to increase the security level of reconfigurable technologies (FPGAs) at the logic, architectural and system levels. The ICTeR project [7] goal was to analyze the potential benefits in terms of security of physical reconfigurable platforms and devices.

As summarized in Table 2, all the competencies necessary for the HardBlare project are available in the consortium. Furthermore, each partner has worked on different fields, thus bringing unique and complementary competencies to the consortium. Furthermore, no partner is unknown to all of the others (i.e., partners have already collaborated in the past at least in couples). This will bring a good collaboration among partners.

Table 2: Consortium expertise

Required competencies	SUPELEC CIDRE	SUPELEC SCEE	UBS MOCS
Processor architecture		X	X
OS-level DIFC	X		
Language-level DIFC	X		
Combining static and dynamic analysis	X		
Hardware coprocessor			X
Dynamic reconfiguration		X	
Embedded systems		X	X

[4] See https://blare-ids.org/

[5] http://www.rennes.supelec.fr/ren/rd/scee/themes/scee_cognitive-radio.html

[6] http://labh-curien.univ-st-etienne.fr/secresoc/doku_wiki/doku.php

[7] http://www.lirmm.fr/~w3mic/ANR/

3 Budget and requested resources

The HardBlare project will involve a total funding of 432K€ as detailed below.

SUPELEC CIDRE

The following permanent resources will be allocated to the project by the SUPELEC CIDRE partner:

- One permanent members of the SUPELEC CIDRE team for about 20% of his (research) time.

The following contribution is requested for this partner (total 164K€):

- 1 software research engineers: 150K€
- Hardware and software (development board + workstation): 6K€
- Travel expenses: 8K€ for (an equivalent of) 3-4 international trips; additional trips will be funded directly by the partner.

SUPELEC SCEE

The following permanent resources will be allocated to the project by the SUPELEC SCEE partner:

- Two permanent members of the SUPELEC SCEE team for about 20% of their (research) time.

The following contribution is requested for this partner (total 164K€):

- 1 hardware research engineers: 150K€
- Hardware and software (development board + workstation): 6K€
- Travel expenses: 8K€ for (an equivalent of) 3-4 international trips; additional trips will be funded directly by the partner.

UBS MOCS

The following permanent resources will be allocated to the project by the UBS MOCS partner:

- Two permanent members of the UBS MOCS team for about 15% of their (research) time.

The following contribution is requested for this partner (total 104K€):

- 1 PhD scholarships: 90K€. The PhD student will be co-supervised by the SUPELEC CIDRE and UBS MOCS teams.
- Hardware and software (development board + workstation): 6K€
- Travel expenses: 8K€ for (an equivalent of) 3-4 international trips; additional trips will be funded directly by the partner.

4 Curriculum Vitae of the principal investigators

SUPELEC CIDRE

Guillaume Hiet : http://www.rennes.supelec.fr/ren/perso/ghiet/

Guillaume Hiet is an associate professor in Computer Science at SUPELEC. He holds engineering diplomas from both SUPELEC and ENSAM, a MSc in Computer Science from SUPELEC and a PhD from University of Rennes. He was formerly a computer security expert at AMOSSYS SAS, a French security consulting company. He was in charge of technical studies, from penetration testing, configuration or vulnerability audit to research and development projects. He worked for various network and system security projects, especially for French national security agencies (ANSSI, DGA). His current research interests are in information flow control and Intrusion Detection Systems. During his PhD, he proposed a DIFC approach to monitor both information flows inside Java application and between different Linux applications. He developed the language-level JBlare DIFC tool. He is also involved in the development of the OS-level Blare DIFC system. He was a member of the program comity of RAID 2011 and RAID 2012 conference.

SUPELEC SCEE

Amor Nafka : http://www.rennes.supelec.fr/ren/perso/anafkha/

Amor NAFKHA received engineering diplomas (M.Sc.) in Electronic and Telecommunication Engineering from the High School of Communications (Sup'Com), Tunis, Tunisia, in 2001. From 2001 to 2003, he rejoined STMicroelectronics as a research engineer and worked on high-level synthesis of STBus (communication architecture developed for System-On-Chip). In 2003, he enrolled at South Brittany University (UBS), Lorient, France where he received the PhD degree in Electronic and Telecommunication Engineering in 2006. From 2006 to 2007, he was a Postdoctoral Fellow at Signal, Communication and Embedded Electronics (SCEE/SUPELEC) Research Team, France. Since 2008, he has been an associate professor at SUPELEC. His research interests include runtime FPGA partial reconfiguration, hardware-accelerated software defined radios, low power design techniques for SOC. Amor NAFKHA has been involved in international and national projects where he has been working on runtime FPGA dynamic partial reconfiguration (European FP6 End-to-end Reconfigurability (E2R), ANR IDROMEL, ANR MOPCOM, DGE TransMedia).

UBS MOCS

Guy Gogniat : http://www-labsticc.univ-ubs.fr/~gogniat/

Guy Gogniat received the Ph.D. degree in electrical and computer engineering from the University of Nice-Sophia Antipolis, France, in 1997. He is currently a Professor in electrical and computer engineering with the University of Bretagne-Sud, Lorient, France, where he has been since 1998. In 2005, he spent one year as an invited Researcher with the University of Massachusetts, Amherst, USA, where he worked on embedded system security using reconfigurable technologies. His work focuses on embedded systems design methodologies and tools. He also conducts research in the domain of reconfigurable and adaptive computing and embedded system security. He is leading a group on embedded system security dealing with communication and data protection, security

overhead reduction, bitstream protection and multiprocessor security. Guy Gogniat has been involved in security since 2003 through different projects. Among them we can mention two projects in collaboration with University of Massachusetts, Amherst USA concerning embedded systems security, monitoring embedded systems and providing countermeasures, one project in collaboration with University of Sao Paulo, Brasil concerning NoC protection and one project in collaboration with ALaRI, Lugano, Switzerland in the domain of embedded security. Guy Gogniat has also been involved in national projects where he has been working on embedded security for FPGAs and on NoC security (DGA SANES, ANR ICTER, ANR SecReSoC). Guy Gogniat is currently leading an ANR project dealing with security in manycore architectures.

References

[1] James P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980. 3

[2] Radoniaina Andriatsimandefitra, Stéphane Geller, and Valérie Viet Triem Tong. Designing information flow policies for android's operating system. In *IEEE International Conference on Communications*, 2012. 3

[3] D.E. Bell and L.J. LaPadula. Secure computer systems: Mathematical foundations. MTR-2547 (ESD-TR-73-278-I) Vol. 1, MITRE Corp., Bedford, 1973. 3, 13

[4] D. F. C. Bewer and M. J. Nash. The chinese wall security policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1989. 3

[5] K. Biba. Integrity considerations for secure computer systems. Technical Report ESD-TR 76-372, MITRE Co., April 1977. 3

[6] Lilian Bossuet, Michael Grand, Lubos Gaspar, Viktor Fischer, and Guy Gogniat. Architectures of flexible symmetric key crypto engines—a survey: From hardware coprocessor to multi-crypto-processor system on chip. *ACM Comput. Surv.*, 45(4):41:1–41:32, August 2013. 13

[7] Deepak Chandra and Michael Franz. Fine-grained information flow analysis and enforcement in a java virtual machine. In *ACSAC*, pages 463–475, 2007. 5, 6

[8] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In *Proceedings of the IEEE Symposium on Security and Privacy (SSP'87)*, pages 184–194. IEEE Society Press, mai 1987. 3

[9] Jérémie Crenne, Romain Vaslin, Guy Gogniat, Jean-Philippe Diguet, Russell Tessier, and Deepak Unnikrishnan. Configurable memory security in embedded systems. *ACM Trans. Embed. Comput. Syst.*, 12(3):71:1–71:23, April 2013. 13

[10] 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*, ISCA '07, pages 482–493, New York, NY, USA, 2007. ACM. 6, 7

[11] Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, 1976. 4

[12] Dorothy E. Denning. An Intrusion-Detection Model. *IEEE transaction on Software Engineering*, 13(2):222–232, 1987. 3

[13] Dorothy E. Denning and Peter J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, July 1977. 4

[14] Petros Efstathopoulos, Maxwell Krohn, Steve VanDeBogart, Cliff Frey, David Ziegler, Eddie Kohler, David Mazières, Frans Kaashoek, and Robert Morris. Labels and event processes in the asbestos operating system. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 17–30, New York, NY, USA, 2005. ACM. 5

[15] Stéphane Geller, Christophe Hauser, Frédéric Tronel, and Valérie Viet Triem Tong. Information flow control for intrusion detection derived from mac policy. In *IEEE International Conference on Communications*, 2011. 5

[16] G. Gogniat, T. Wolf, W. Burleson, J-P Diguet, L. Bossuet, and R. Vaslin. Reconfigurable hardware for high-security/ high-performance embedded systems: The safes perspective. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(2):144–155, 2008. 13

[17] J. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symposium on Research in Security and Privacy*, 1982. 4

[18] Gurvan Le Guernic and Thomas Jensen. Monitoring information flow. In Andrei Sabelfeld, editor, *Proceedings of the Workshop on Foundations of Computer Security*, pages 19–30. DePaul University, JUN 2005. 6

[19] William G. J. Halfond, Alessandro Orso, and Panagiotis Manolios. Using positive tainting and syntax-aware evaluation to counter sql injection attacks. In *SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 175–185, New York, NY, USA, 2006. ACM. 5, 6

[20] William R. Harris, Somesh Jha, and Thomas Reps. Difc programs by automatic instrumentation. In *CCS '10: Proceedings of the 17th ACM conference on Computer and communications security*, pages 284–296, New York, NY, USA, 2010. ACM. 5

[21] C. Hauser, F. Tronel, J. Reid, and C. Fidge. A taint marking approach to confidentiality violation detection. In C. Pieprzyk, J.and Thomborson, editor, *Australasian Information Security Conference (AISC 2012)*, volume 125 of *CRPIT*, pages 83–90, Melbourne, Australia, 2012. ACS. 5

[22] G. Hiet, L. Mé, B. Morin, and V. Viet Triem Tong. Monitoring both os and program level information flows to detect intrusions against network servers. In *IEEE Workshop on "Monitoring, Attack Detection and Mitigation"*, 2007. 5, 6, 13

[23] Guillaume Hiet, Valerie Viet Triem Tong, Ludovic Me, and Benjamin Morin. Policy-based intrusion detection in web applications by monitoring java information flows. *Int. J. Inf. Comput. Secur.*, 3(3/4):265–279, 2009. 5, 13

[24] James A. Hoagland, Raju Pandey, and Karl N. Levitt. Security policy specification using a graphical approach. Technical report, University of California, Davis, 1998. 3

[25] H. Kannan, M. Dalton, and C. Kozyrakis. Decoupling dynamic information flow tracking with a dedicated coprocessor. In *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, pages 105–114, 2009. 7, 8

[26] Hari Kannan. Ordering decoupled metadata accesses in multiprocessors. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 381–390, New York, NY, USA, 2009. ACM. 7

[27] Maxwell Krohn, Alexander Yip, Micah Brodsky, Natan Cliffer, M. Frans Kaashoek, Eddie Kohler, and Robert Morris. Information flow control for standard os abstractions. In *Proceedings of the 21st Symposium on Operating Systems Principles*, Stevenson, WA, October 2007. 5

[28] Lap Chung Lam and Tzi cker Chiueh. A general dynamic information flow tracking framework for security applications. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, pages 463–472, Washington, DC, USA, 2006. IEEE Computer Society. 5, 6

[29] Peter Loscocco and Stephen Smalley. Integrating flexible support for security policies into the linux operating system. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01)*, 2001. 5

[30] Florent Marchand de Kerchove, Jacques Noyé, and Mario Südholt. Aspectizing javascript security. In *Proceedings of the 3rd Workshop on Modularity in Systems Software*, MISS '13, pages 7–12, New York, NY, USA, 2013. ACM. 5

[31] Andrew C. Myers. Jflow: Practical mostly-static information flow control. In *Proceedings of the 26th ACM on Principles of Programming Languages*, 1999. 4

[32] Andrew C. Myers and Barbara Liskov. A decentralized model for information flow control. *SIGOPS Oper. Syst. Rev.*, 31(5):129–142, 1997. 4

[33] Srijith K. Nair, Patrick N. D. Simpson, Bruno Crispo, and Andrew S. Tanenbaum. A virtual machine based information flow control system for policy enforcement. In *First International Workshop on Run Time Enforcement for Mobile and Distributed Systems (REM 2007)*, pages 1–11, Dresden, Germany, 2007. 5, 6

[34] James Newsome and Dawn Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS 2005)*, San Diego, CA, February 2005. 5, 6

[35] Anh Nguyen-Tuong, Salvatore Guarnieri, Doug Green, , Jeffrey Shirley, and David Evans. Automatically hardening web applications using precise tainting. In *IFIP Security Conference*, 2005. 5, 6

[36] François Pottier and Vincent Simonet. Information flow inference for ml. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2003. 4

[37] Feng Qin, Cheng Wang, Zhenmin Li, Ho seop Kim, Yuanyuan Zhou, and Youfeng Wu. Lift: A low-overhead practical information flow tracking system for detecting security attacks. *39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006. 5, 6

[38] Indrajit Roy, Donald E. Porter, Michael D. Bond, Kathryn S. McKinley, and Emmett Witchel. Laminar: practical fine-grained decentralized information flow control. In *Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation*, 2009. 6

[39] John Rushby. Noninterference, transitivity and channel-control security policies. Technical report, SRI, 1992. 3

[40] A Sabelfeld and A C Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1), 2003. 4, 5

[41] Andrei Sabelfeld and Alejandro Russo. From dynamic to static and back: Riding the roller coaster of information-flow control research. In *Perspectives of Systems Informatics*, volume 5947. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 4

[42] Delphine Demange Suraj Iyerx Aleksey Kligerx Greg Morrisetty Benjamin C. Pierce Howard Reubensteinx Jonathan M. Smith Gregory T. Sullivanx Arun Thomasx Jesse Tovy Christopher M. Whitex David Wittenbergx Silviu Chiricescux, Andre DeHon. Safe: A clean-slate architecture for secure systems. 2013. 2, 7

[43] G. Edward Suh, Jae W. Lee, David Zhang, and Srinivas Devadas. Secure program execution via dynamic information flow tracking. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XI, pages 85–96, New York, NY, USA, 2004. ACM. 7

[44] Guru Venkataramani, Ioannis Doudalis, Yan Solihin, and Milos Prvulovic. Flexitaint: A programmable accelerator for dynamic taint propagation. In *In 14th International Symposium on HighPerformance Computer Architecture (HPCA-14*, 2008. 7, 8

[45] Dennis Volpano, Cynthia Irvine, and Geoffrey Smith. A sound type system for secure flow analysis. *Journal in Computer Security*, 4(2-3):167–187, 1996. 4

[46] Robert Watson and Chris Vance. The trustedbsd mac framework: Extensible kernel access control for freebsd 5.0. In *In USENIX Annual Technical Conference*, pages 285–296, 2003. 5

[47] Wei Xu, Sandeep Bhatkar, and R. Sekar. Taint-enhanced policy enforcement: a practical approach to defeat a wide range of attacks. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, pages 9–9, Berkeley, CA, USA, 2006. USENIX Association. 5, 6

[48] Steve Zdancewic. Challenges for information-flow security. In *Proceedings of the 1st International Workshop on Programming Language Interference and Dependence*, 2004. 4

[49] Nickolai Zeldovich, Silas Boyd-Wickizer, Eddie Kohler, and David Mazières. Making information flow explicit in histar. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 263–278, Berkeley, CA, USA, 2006. USENIX Association. 5

[50] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. Experimenting with a policy-based hids based on an information flow control model. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, December 2003. 13

[51] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. An improved reference flow control model for policy-based intrusion detection. In *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS)*, October 2003. 3, 13