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:

HAL Id: hal-01311032
https://hal-centralesupelec.archives-ouvertes.fr/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 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 `x`.

**Example code**

```
p = 3;
p ← public
s = 42;
s ← secret
x = p + s;
if (x != public)
    raise interruption
print(x);
```

### Introduction

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, Vianny Lapître 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

### State of the art

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

### Static Analysis

**OFFLINE**

1. Source code compilation
2. LLVM IR generation
3. Static Analysis
4. ELF File

**ONLINE**

1. ARM CoreSight Components export trace (for both CPUs) towards PL in PFT
2. ARM Cortex-A9 Trace mode: Coresight components
   - Processing System (PS)
   - FPGA Programmable Logic (PL)
   - 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 tags 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

**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**

1. Trace: address of `X`
2. `X = tag_off(“password.txt”)`
3. `Y = tag_off(“foo.txt”)`
4. `Buffer = X`
5. `Buffer = Y`
6. `Socket = Buffer + Y`

**Some References**