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.
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;
if (x + s ← s Tag propagation
  if (s != public)
    Tag check
      x = p + s;
      s = 42;
      print(x);
```

### Introduction

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

Mounir Nasr Allah, Guillaume Hiet, Muhammad Abdul Wahab, Pascal Cotret, Guy Gogniat, Via Vanney Lapôtre

© CIDRE / IRISA / INRIA, CentraleSupélec, Cesson-Sévigné - FRANCE
© SCEE / IETR, CentraleSupélec, Cesson-Sévigné - FRANCE
© Lab-STICC, University of South Brittany, Lorient - FRANCE

### State of the art

<table>
<thead>
<tr>
<th></th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Flexible security policies</td>
<td>Overhead</td>
</tr>
<tr>
<td></td>
<td>Multiple attacks detected</td>
<td>(from 300%) to (3700%)</td>
</tr>
<tr>
<td>Hardware</td>
<td>Low overhead (&lt;10%)</td>
<td>Invasive modifications</td>
</tr>
<tr>
<td>In-core DIFT</td>
<td>Low overhead (&lt;10%)</td>
<td>Few security policies</td>
</tr>
<tr>
<td>Dedicated CPU for DIFT</td>
<td>Low overhead (&lt;10%)</td>
<td>Few modifications to CPU Energy consumption (x 2)</td>
</tr>
<tr>
<td>Dedicated DIFT Coprocessor</td>
<td>Flexible security policies</td>
<td>Communication between CPU and DIFT Coprocessor</td>
</tr>
</tbody>
</table>

###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

### 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
- 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/multi-core systems.