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 codeign 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 as tags) to denote the type of information that are saved or generated within the system.

**Example code**

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

<table>
<thead>
<tr>
<th>Example code</th>
<th>Tag initialization</th>
<th>Tag propagation</th>
<th>Tag check</th>
</tr>
</thead>
<tbody>
<tr>
<td>p = 3;</td>
<td>p ← public</td>
<td>s ← secret</td>
<td>x ← p + s = 5</td>
</tr>
<tr>
<td>s = 42;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p = 3;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s = p + s;</td>
<td></td>
<td></td>
<td>if (x != public)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>raise interruption</td>
</tr>
</tbody>
</table>

### State of the art

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
</tr>
<tr>
<td>In-core DIFT</td>
<td></td>
</tr>
<tr>
<td>Dedicated CPU for DIFT</td>
<td></td>
</tr>
<tr>
<td>Dedicated DIFT Coprocessor</td>
<td></td>
</tr>
</tbody>
</table>

#### Hybrid

- **Flexible security policies**
- **Overhead** (from 300% to 3700%)
- **Invasive modifications**
- **Fixed security policies**
- **Invasive modifications**
- **Wasting resources**
- **Energy consumption (x 2)**
- **Communication between CPU and DIFT Coprocessor**
- **CPU not modified**

### Static Analysis

- **Flexible security policies**
- **Overhead**
- **Invasive modifications**
- **Wasting resources**
- **Energy consumption (x 2)**
- **Communication between CPU and DIFT Coprocessor**
- **CPU not modified**

### Dynamic Information Flow Tracking (DIFT)

- **Dedicated DIFT Coprocessor**
- **Low overhead (<10%)**
- **CPU not modified**
- **Energy consumption (x 2)**
- **Communication between CPU and DIFT Coprocessor**

### Dynamic Information Flow Tracking (DIFT) step-by-step

1. **Operating System (bBlare)**
2. **Flexitaint**
3. **Flexitaint**
4. **Flexitaint**
5. **Flexitaint**
6. **Flexitaint**

#### Definitions

- **Tag dependencies**
- **Memory tags**
- **Tag register file**

#### DIFT Coprocessor

- **Low overhead (<10%)**
- **CPU not modified**
- **Energy consumption (x 2)**
- **Communication between CPU and DIFT Coprocessor**

### ARM Cortex-A9 Trace mode: Coresight components

- **Processing System (PS)**
- **Programmable Logic (PL)**
- **Traces**
- **PTF Decoder**
- **DIFT Coprocessor**
- **Tag Register File**
- **Memory Tags**
- **Tag Dependencies**

### 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/manycore systems.**

---


[3] H. Kannan, M. Dalton, and C. Kozyrakis, “Decoupling dynamic information flow tracking with a dedicated core,” Program Trace Macrocell (PTM) generates a trace containing the address for each committed instruction modifying the PC value.
