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 (not public).

### Example code
```
print(x);
p = 3;
s = 42;
x = p + s;
```

### Example of a program flow
```
if (x != public) raise interruption
```

### Towards a hardware-assisted DIFT system

#### Advantages
- **Flexible security policies**
- **Multiple attacks detected**
- **Overhead** (from 300% to 3700%)

#### Disadvantages
- **Invasive modifications**

### State of the art

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

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

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

#### 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

### Some References