PAGURUS: Low-Overhead Dynamic Information Flow Tracking on Loosely Coupled Accelerators

Luca Piccolboni, Giuseppe Di Guglielmo and Luca P. Carloni
Columbia University, NY, USA
Systems-on-Chip (SoCs) Are Vulnerable to Software Attacks

[۱۷] M. Gautschi et al., IEEE VLSI ’17

[M. Gautschi et al., IEEE VLSI ’17]
Attacking PULPino
Buffer-Overflow Attack

memory location: 0xAA

```c
int buff[10], k;
int (*fun)(int) = foo;
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
for (k = 0; k < num; ++k)
    buff[k] = sw(val);
fun(1);  // call foo?
```

<table>
<thead>
<tr>
<th>val</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>10</td>
</tr>
<tr>
<td>fun</td>
<td>0xAA</td>
</tr>
<tr>
<td>buff[0]</td>
<td>sw(7)</td>
</tr>
<tr>
<td>buff[1]</td>
<td>sw(7)</td>
</tr>
<tr>
<td>buff[9]</td>
<td>sw(7)</td>
</tr>
</tbody>
</table>

main memory

Buffer Overflow Attack
Attacking PULPino
Buffer-Overflow Attack

```c
int buff[10], k;
int (*fun)(int) = foo;
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
for (k = 0; k < num; ++k)
  buff[k] = sw(val);
fun(1); // call foo?
```

memory location: 0xAA

can be used to call a malicious function

<table>
<thead>
<tr>
<th>val = 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>num = 11</td>
</tr>
<tr>
<td>fun = sw(7)</td>
</tr>
<tr>
<td>buff[9] = sw(7)</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>buff[1] = sw(7)</td>
</tr>
<tr>
<td>buff[0] = sw(7)</td>
</tr>
</tbody>
</table>

main memory
Attacking PULPino

Dynamic Information Flow Tracking (DIFT)

```c
int buff[10], k;
int (*fun)(int) = foo;
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
for (k = 0; k < num; ++k)
    buff[k] = sw(val);
fun(1); // call fun
```

memory location: 0xAA

<table>
<thead>
<tr>
<th>val</th>
<th>num</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>fun</td>
<td>7</td>
</tr>
<tr>
<td>buff[9]</td>
<td>7</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>buff[1]</td>
<td>7</td>
</tr>
<tr>
<td>buff[0]</td>
<td>7</td>
</tr>
</tbody>
</table>

G. E. Suh et al., ACM ASPLOS ’04
Homogenous SoCs
Now Secured with DIFT

- Processor Core (RI5CY)
- Data RAM
- Instr. RAM
- Boot. RAM
- UART
- SPI M.

DIFT Extensions

[M. Gautschi et al., IEEE VLSI ’17]
[C. Palmiero et al., IEEE HPEC ’18]
Heterogeneous SoCs
No-More-Secured with DIFT

- Processor Core (RI5CY)
- Data RAM
- Instr. RAM
- Boot. RAM
- UART
- SPI M.
- Loosely Coupled Accelerator #1
- Loosely Coupled Accelerator #2

[DIFT Extensions]
[M. Gautschi et al., IEEE VLSI ’17]
[C. Palmiero et al., IEEE HPEC ’18]
Attacking PULPino (Again)
Buffer-Overflow Attack

```c
int buff[10] = {0};
int (*f)(int) = foo;
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
hw(num, val, buff);
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>main memory</td>
<td>tags</td>
<td></td>
</tr>
<tr>
<td>val = 7</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>num = 11</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>fun = 0xAA</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>buff[9] = 0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>buff[1] = 0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>buff[0] = 0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Attacking PULPino (Again)
Buffer-Overflow Attack

```c
int buff[10] = {0};
int (*f)(int) = foo;
int num = atoi(argv[1]);
int val = atoi(argv[2]);
/* this is a bad idea */
hw(num, val, buff);
```

The accelerator is not able to propagate the tags.
Contributions

1. We propose PAGURUS, a methodology to design a circuit shell that adds DIFT support to accelerators
Contributions

PULPino System-on-Chip

Processor Core (RI5CY)  Data RAM  Instr. RAM  Boot. RAM

Loosely Coupled Accelerator #1  Loosely Coupled Accelerator #2

UART  SPI M.
Contributions

1. We propose PAGURUS, a methodology to design a circuit shell that adds DIFT support to accelerators
   a) The shell design is *independent* from the design of the accelerators and vice versa
   b) The shell has *low overheads* on both the performance and cost of accelerators

2. We propose a *metric* to quantitatively measure the security guarantees provided by the shell
Preliminaries
Assumptions and Attack Model

1. The hardware is safe: no hardware Trojans

2. The software is not safe: it contains bugs and vulnerabilities useful for the attackers

The attackers exploit these vulnerabilities through common I/O interfaces with the goal of affecting the integrity and/or the confidentiality of the hardware-accelerated software applications
## Preliminaries

### Tagging Scheme

<table>
<thead>
<tr>
<th>main memory</th>
<th>tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>value #1</td>
<td>tag #1</td>
</tr>
<tr>
<td>value #2</td>
<td>tag #2</td>
</tr>
<tr>
<td>value #3</td>
<td>tag #3</td>
</tr>
</tbody>
</table>

[J. Porquet et al., ACM/IEEE CODES’13]

1. Coupled Scheme
Preliminaries

Tagging Scheme

1. Coupled Scheme

2. Decoupled Scheme

[J. Porquet et al., ACM/IEEE CODES’13]
Preliminaries
Tagging Scheme

1. Coupled Scheme

2. Decoupled Scheme
   2.1. Interleaved Scheme

\textbf{tag offset} = \# words in memory between two consecutive values

\[\text{(tag offset} = 1)\]

[J. Porquet et al., ACM/IEEE CODES’13]
Contributions

1. We propose **PAGURUS**, a methodology to design a circuit shell that adds DIFT support to accelerators
   
   a) The shell design is *independent* from the design of the accelerators and vice versa
   
   b) The shell has *low overheads* on both the performance and cost of accelerators
Accelerators Architecture

Loosely Coupled Accelerator

- main memory
- register #1
- register #2
- ... (up to register #K)

Configuration

- reg #1
- ... (up to reg #K)

Private local memory / scratchpad

- bank
- bank
- bank
- bank
Accelerators Architecture

Loosely Coupled Accelerator

- Configuration
  - reg #1 ...
  - reg #K

- Load input
  - val val val

- Memory
  - banks
  - Private local memory / scratchpad
Accelerators

Architecture

main memory

Loosely Coupled Accelerator

configuration
reg #1 ... reg #K

load input
val val val

compute
val val val

bank bank bank bank

private local memory / scratchpad
Accelerators Architecture

Loosely Coupled Accelerator

- Configuration
  - Configuration registers: `reg #1` to `reg #K`

- Load input
  - Input values: `val` repeated

- Compute
  - Output: `val` repeated

- Store output
  - Output: `val` repeated

- Private local memory / scratchpad
  - Memory banks

- Burst length
- Output
- Main memory
DIFT Shell Architecture

Loosely Coupled Accelerator

Accelerator
DIFT Shell
Architecture

main memory

register #1
register #2
...
register #K
reg. #K+1: src_tag
reg. #K+2: dst_tag

shell configuration
src_tag
dst_tag

Loosely Coupled Accelerator
Accelerator
DIFT Shell Architecture

- Burst length
- src_tag
- src_tag
- Input
- Main memory

Shell configuration:
- src_tag
- dst_tag

Shell load logic:
- val
- tag
- val
- tag

if tag != src_tag
DIFT_exception!
DIFT Shell Architecture

burst length
dst_tag
dst_tag
output
main memory

shell configuration
src_tag dst_tag

shell load logic
val tag val tag

shell store logic
val tag val tag

accelerator
val val

Loosely Coupled Accelerator
Contributions

1. We propose PAGURUS, a methodology to design a circuit shell that adds DIFT support to accelerators
   a) The shell design is *independent* from the design of the accelerators and vice versa
   b) The shell has *low overheads* on both the performance and cost of accelerators

2. We propose a *metric* to quantitatively measure the security guarantees provided by the shell
A Security Metric
Definition

value #1
value #2
src_tag
value #3
input
main memory
A Security Metric

Definition

<table>
<thead>
<tr>
<th>Value #1 [overwritten]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value #2 [overwritten]</td>
</tr>
<tr>
<td>src_tag [overwritten]</td>
</tr>
<tr>
<td>Value #3 [overwritten]</td>
</tr>
<tr>
<td>Input</td>
</tr>
<tr>
<td>Value #1</td>
</tr>
<tr>
<td>Output</td>
</tr>
<tr>
<td>Main memory</td>
</tr>
</tbody>
</table>

Loosely Coupled Accelerator

DIFT Shell

val val val tag
val tag val val

ACM/IEEE CODES + ISSS 2018, Turin, Italy
A Security Metric

Definition

Information Leakage

- *Quantitative* metric for security
A Security Metric

Analysis

• **Information Leakage:** amount of data that can be produced as output by an accelerator before its shell realizes that the input has been corrupted

1. Tag offset: $\uparrow$ tag offset $\uparrow$ leakage
2. Algorithm: $\downarrow$ I/O ratio $\uparrow$ leakage

I/O ratio: the number of load bursts necessary to produce a store burst
A Security Metric

Analysis

- **Information Leakage**: amount of data that can be produced as output by an accelerator before its shell realizes that the input has been corrupted

1. Tag offset: ↑ tag offset ↑ leakage
2. Algorithm: ↓ I/O ratio ↑ leakage
3. Implementation: ↑ burst len. ↓ leakage
4. Workload: ↓ work. size ↓ leakage
Experimental Results
Experimental Setup (1/2)

• We designed three loosely coupled accelerators:
  • GRAY: converts a RGB image into a grayscale image
  • MEAN: calculates the mean of a 2D matrix (columns)
  • MULTS: multiplies a 2D matrix by its transpose
Experimental Results

Experimental Setup (1/2)

• We designed three loosely coupled accelerators:
  • GRAY: converts a RGB image into a grayscale image
  • MEAN: calculates the mean of a 2D matrix (columns)
  • MULTS: multiplies a 2D matrix by its transpose
Experimental Results

Experimental Setup (1/2)

• We designed three loosely coupled accelerators:
  • GRAY: converts a RGB image into a grayscale image
  • MEAN: calculates the mean of a 2D matrix (columns)
  • MULTS: multiplies a 2D matrix by its transpose
Experimental Results
Experimental Setup (1/2)

- We designed three loosely coupled accelerators:
  - **GRAY**: converts a RGB image into a grayscale image
  - **MEAN**: calculates the mean of a 2D matrix (columns)
  - **MULTS**: multiplies a 2D matrix by its transpose
Experimental Results
Experimental Setup (1/2)

- We designed three loosely coupled accelerators:
  - GRAY: converts a RGB image into a grayscale image
  - MEAN: calculates the mean of a 2D matrix (columns)
  - MULTS: multiplies a 2D matrix by its transpose

- We designed the accelerators and the shell in SystemC

- We used *Cadence Stratus HLS* for high-level synthesis and *Xilinx Vivado* for logic synthesis → Virtex-7 FPGA
Experimental Results
Experimental Setup (2/2)

We explored different alternatives by varying:

- accelerator
- tag offset
- burst size
- workload
  - 128 x 128 - small
  - 512 x 512 - medium
  - 2048 x 2048 - large

\[\text{[P. Mantovani et al., ACM/IEEE DAC '16]}\]
\[\text{[L. P. Carloni, ACM/IEEE DAC '16]}\]
Experimental Results

Quantitative Security Analysis - MEAN

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results

Quantitative Security Analysis - MEAN

- max information leakage => the highest tag offset
- min information leakage => the lowest tag offset
Experimental Results
Quantitative Security Analysis - MEAN

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results

Quantitative Security Analysis - GRAY

![Graph showing information leakage vs burst size for medium, small, and large sizes.](#)
Experimental Results

Quantitative Security Analysis - GRAY

- Information leakage (%)
- Burst size (bytes)

medium
- $2^{19}$
- $2^5$
- $2^{15}$
- $2^5$

small
- $2^{19}$
- $2^5$
- $2^{15}$
- $2^5$

large
- $2^{23}$

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results
Quantitative Security Analysis - MULTS

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results

Quantitative Security Analysis - MULTS

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results

Performance Analysis - GRAY

- medium
- large
- no tags

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results
Performance Analysis - GRAY

 normalized execution time
burst size (bytes)

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results

Performance Analysis - GRAY

normalized execution time vs. burst size (bytes)

- large
- medium
- small

no tags

$2^0$

$2^6$

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Experimental Results

Performance Analysis - GRAY

ACM/IEEE CODES + ISSS 2018, Turin, Italy
Conclusions

• We propose PAGURUS, a flexible methodology to design a **shell** that extends DIFT to accelerators
  
  1. The shell design is independent from the accelerator design and vice versa
  
  2. The shell has negligible cost overhead and reasonable performance overhead

• We define the **metric** of information leakage for accelerators to quantitatively measure security
PAGURUS: Low-Overhead Dynamic Information Flow Tracking on Loosely Couple Accelerators

Questions?

Speaker: Luca Piccolboni
Columbia University, NY