# Intrusion detection through monitoring of timing deviations

**Nicolas BELLEC**

ENS Rennes - Univ Rennes 1

June 25, 2019

Supervisor : Isabelle Puaut, Simon Rokicki

UNIVERSITÉ DE RENNES 1

# Motivation

- Real-time embedded systems

# Motivation

- Real-time embedded systems

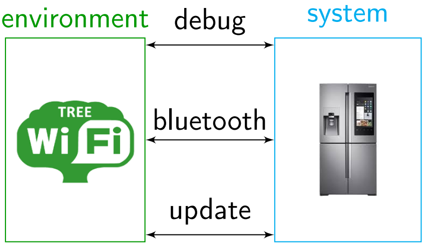
# Motivation

- Real-time embedded systems

# Motivation

- Real-time embedded systems

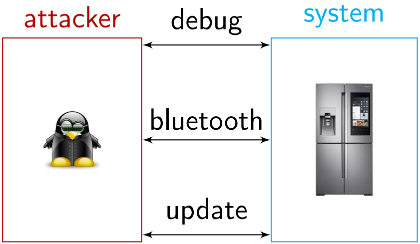
- Openness ...



# Motivation

- Real-time embedded systems

- Openness … leads to more vulnerabilities



# Context - Real Time

Real-time = Guarantee response within specified time constraints

# Context - Real Time

Real-time = Guarantee response within specified time constraints

- WCET = Worst-Case Execution Time [1]

1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

# Context - Real Time

Real-time = Guarantee response within specified time constraints

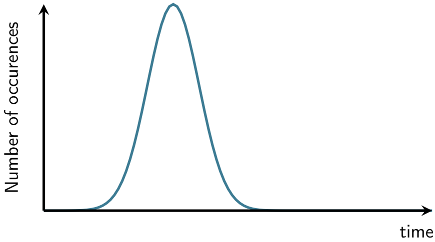
- WCET = Worst-Case Execution Time [1]
  - Isolated task
  - On a binary/hardware pair

1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

# Context - Real Time

Real-time = Guarantee response within specified time constraints

- WCET = Worst-Case Execution Time [1]
  - Isolated task
  - On a binary/hardware pair

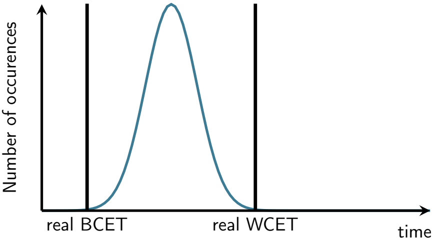



1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

# Context - Real Time

Real-time = Guarantee response within specified time constraints

- WCET = Worst-Case Execution Time [1]
  - Isolated task
  - On a binary/hardware pair

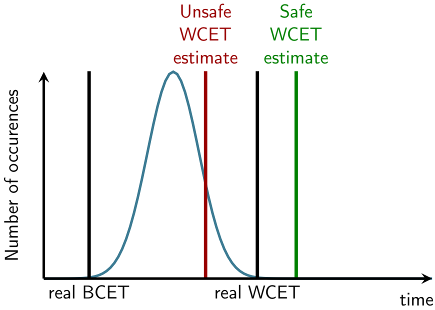



1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools". In: ACM Trans. Embedded Comput. Syst. 2008

# Context - Real Time

Real-time = Guarantee response within specified time constraints

- WCET = Worst-Case Execution Time [1]
  - Isolated task
  - On a binary/hardware pair

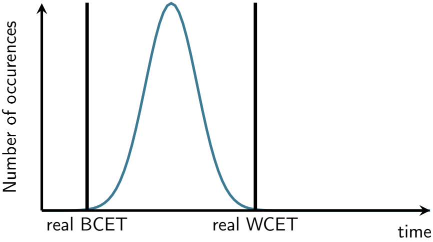



1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

# Context - Real Time

Real-time = Guarantee response within specified time constraints

- WCET = Worst-Case Execution Time [1]
  - Isolated task
  - On a binary/hardware pair

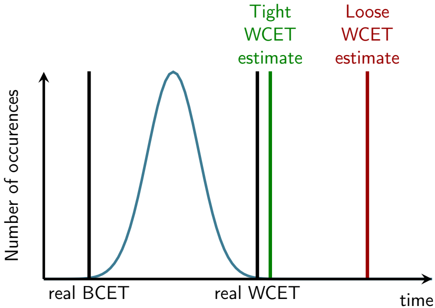



1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

# Context - Real Time

Real-time = Guarantee response within specified time constraints

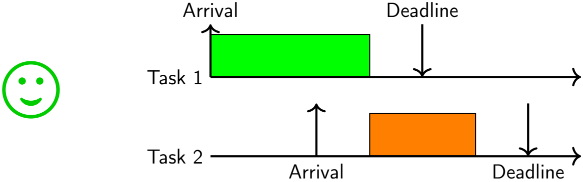
- WCET = Worst-Case Execution Time [1]
  - Isolated task
  - On a binary/hardware pair




1. Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

# Real-time - Schedule [2]

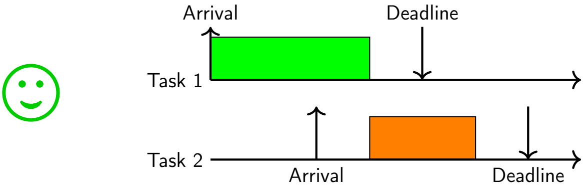
- Scheduling = Combine tasks WCETs and system requirements




2. Buttazzo, G. "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications"

# Real-time - Schedule [2]

- Scheduling = Combine tasks WCETs and system requirements

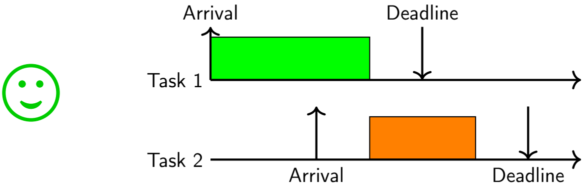



but sometimes · · ·

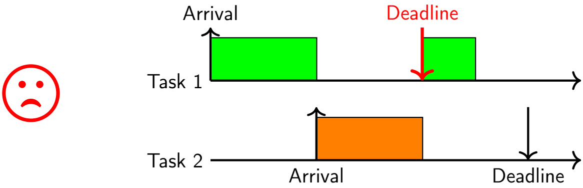
2. Buttazzo, G. "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications"

# Real-time - Schedule [2]

- Scheduling = Combine tasks WCETs and system requirements




but sometimes · · ·




2. Buttazzo, G. "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications"

# Security - Control-flow hijacking

Goal : Modify the program behavior to execute arbitrary code

# Security - Control-flow hijacking

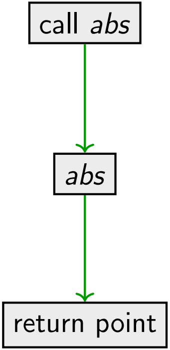
Goal : Modify the program behavior to execute arbitrary code

- Principle

# Security - Control-flow hijacking

Goal : Modify the program behavior to execute arbitrary code

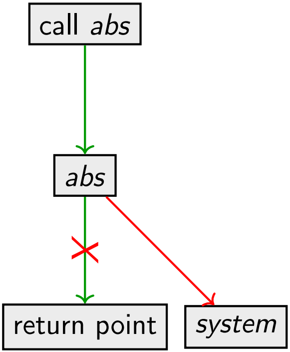
- Principle



# Security - Control-flow hijacking

Goal : Modify the program behavior to execute arbitrary code

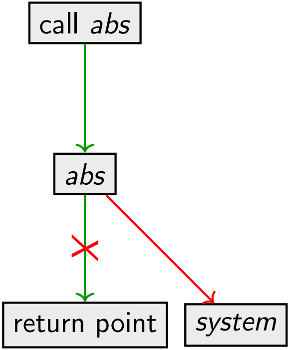
- Principle



# Security - Control-flow hijacking

Goal : Modify the program behavior to execute arbitrary code

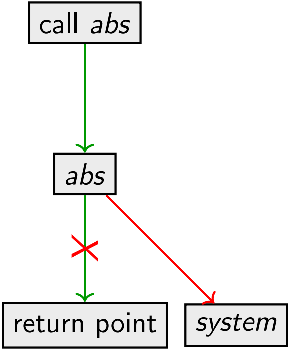
- Principle




- Example

Goal : Modify the program behavior to execute arbitrary code

- Principle



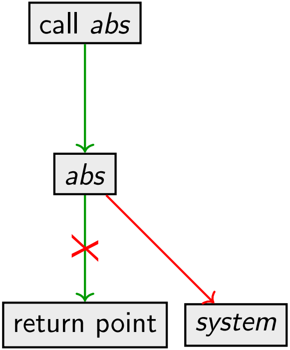

- Example

| buffer | i | retval |
|---|---|---|

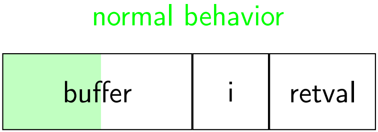
# Security - Control-flow hijacking

Goal : Modify the program behavior to execute arbitrary code

- Principle



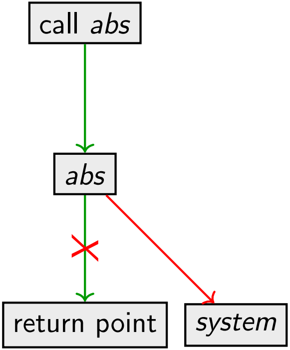

- Example



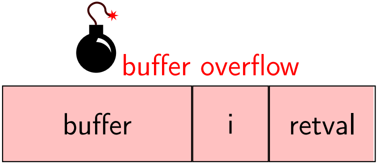
# Security - Control-flow hijacking

Goal : Modify the program behavior to execute arbitrary code

- Principle




- Example



# State of the art

- Diversity techniques
  - Schedule level diversity [3]
  - Program level diversity [4]

- Diversity techniques
  - Schedule level diversity [3]
  - Program level diversity [4]

- Monitoring techniques

# State of the art

- Diversity techniques
  - Schedule level diversity [3]
  - Program level diversity [4]

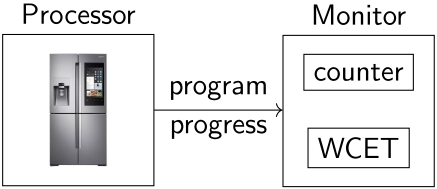
- Monitoring techniques

| Name | Domain | Information | Overhead / Intrusiveness |
|---|---|---|---|
| T-Rex[5] | RT | Checkpoint / WCET | High |
| T-ProT[5] | RT | Checkpoint / WCET | High |
| T-AxT[5] | RT | Time/Space correlation | High |
| Yoon et al.[6] | RT | Syscall learning | Small |
| Chevalier et al.[7] | G | Control-Flow | Small |

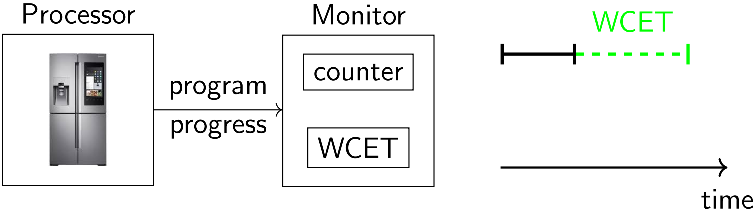
# Contribution

# Contribution

- Hardware monitor (0 overhead)

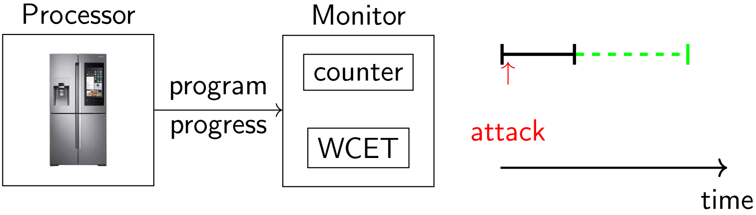


- Hardware monitor (0 overhead)

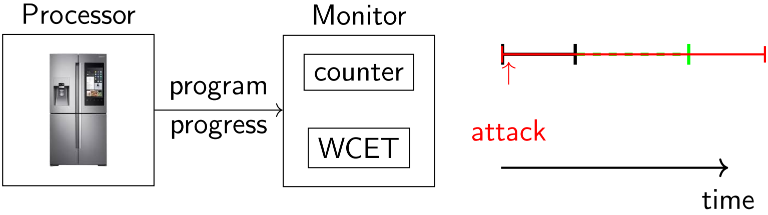


# Contribution

- Hardware monitor (0 overhead)

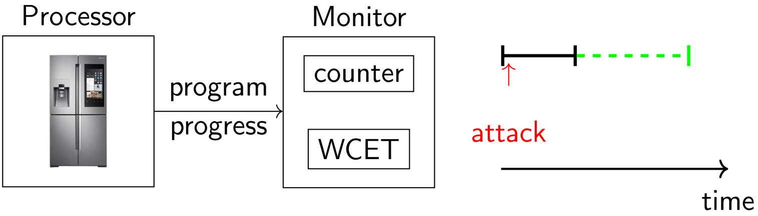


- Hardware monitor (0 overhead)
- Detecting attacks that exceed the WCET (limited threshold)



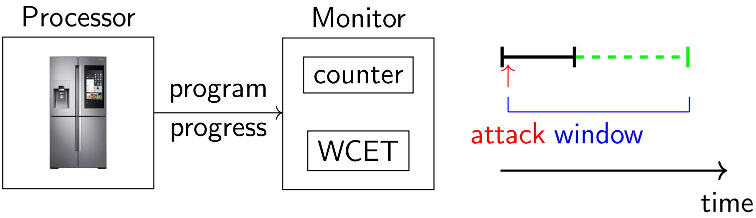
# Contribution

- Hardware monitor (0 overhead)
- Detecting attacks that exceed the WCET (limited threshold)



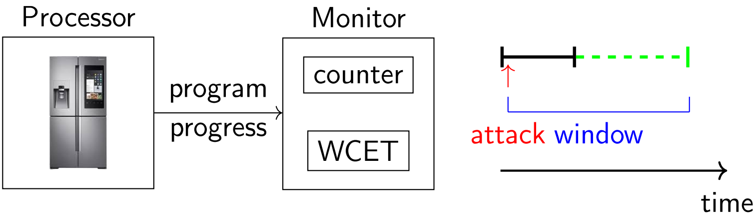
# Contribution

- Hardware monitor (0 overhead)
- Detecting attacks that exceed the WCET (limited threshold)



# Contribution

- Hardware monitor (0 overhead)
- Detecting attacks that exceed the WCET (limited threshold)




- Attack scenario : Control-Flow Hijacking

# Outline

- Region definition
- Automatic region selection
- Monitoring example
- Experiments
- Future Work

# Regions of interest

- *Single Entry Single Exit* (SESE) regions [3]

3. Richard Johnson et al. "The program structure tree: computing control regions in linear time". In: ACM SIGPLAN 1994 conference on Programming language design and implementation

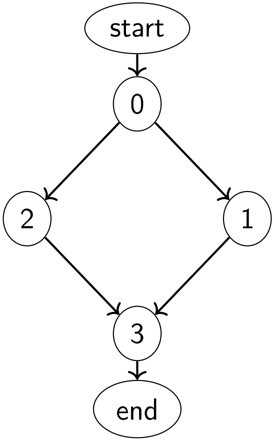
# Regions of interest

- *Single Entry Single Exit* (SESE) regions [3]
  - One entry point
  - One exit point
  - Self contained

3. Richard Johnson et al. "The program structure tree: computing control regions in linear time". In: ACM SIGPLAN 1994 conference on Programming language design and implementation

# Regions of interest

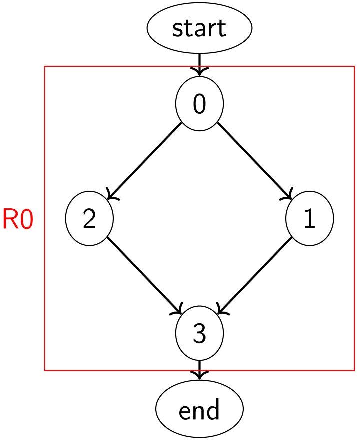
- *Single Entry Single Exit* (SESE) regions [3]
  - One entry point
  - One exit point
  - Self contained




3. Richard Johnson et al. "The program structure tree: computing control regions in linear time". In: ACM SIGPLAN 1994 conference on Programming language design and implementation

# Regions of interest

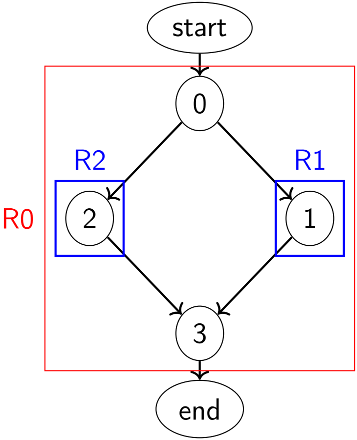
- *Single Entry Single Exit* (SESE) regions [3]
  - One entry point
  - One exit point
  - Self contained




3. Richard Johnson et al. "The program structure tree: computing control regions in linear time". In: ACM SIGPLAN 1994 conference on Programming language design and implementation

# Regions of interest

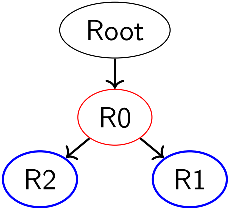
- *Single Entry Single Exit* (SESE) regions [3]
  - One entry point
  - One exit point
  - Self contained

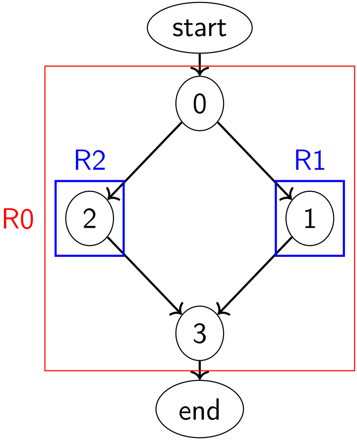



3. Richard Johnson et al. "The program structure tree: computing control regions in linear time". In: ACM SIGPLAN 1994 conference on Programming language design and implementation

# Regions of interest

- *Single Entry Single Exit* (SESE) regions [3]
  - One entry point
  - One exit point
  - Self contained
  - Simple nesting



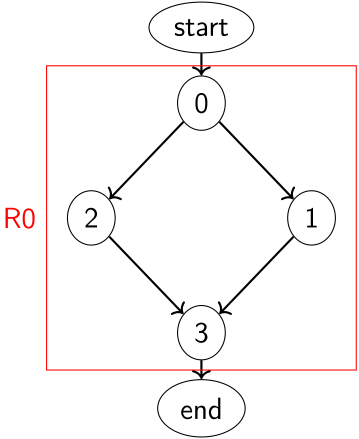

3. Richard Johnson et al. "The program structure tree: computing control regions in linear time". In: ACM SIGPLAN 1994 conference on Programming language design and implementation

# Maximal Inner Time (MIT)

The time of a region without the time of monitored sub-regions

# Maximal Inner Time (MIT)

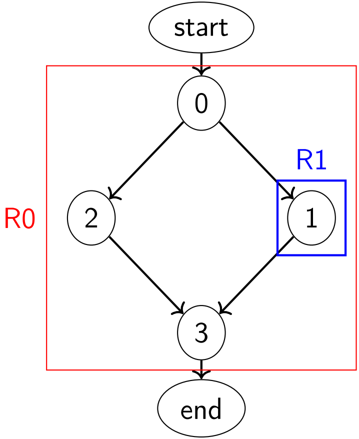
The time of a region without the time of monitored sub-regions




MIT R0 =
WCET R0

# Maximal Inner Time (MIT)

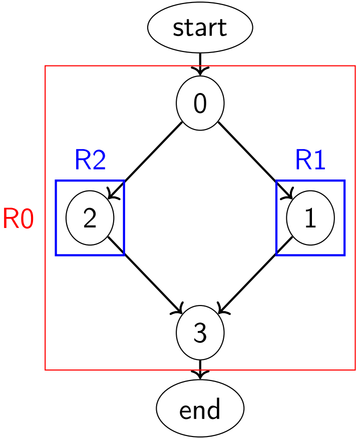
The time of a region without the time of monitored sub-regions




MIT R0 =
WCET ( R0 \ R1 )

# Maximal Inner Time (MIT)

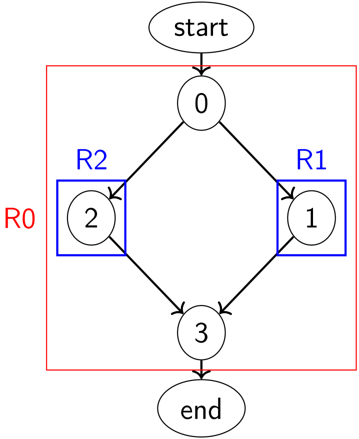
The time of a region without the time of monitored sub-regions




MIT R0 =
WCET ( R0 \ { R1, R2 })

# Maximal Inner Time (MIT)

The time of a region without the time of monitored sub-regions




MIT R0 =
WCET ( R0 \ { R1, R2 })

MIT = monitor threshold

# SESE Selection Problem (SSP)

- Constraints

- Memory size
- Stack size
- Comparison per cycle

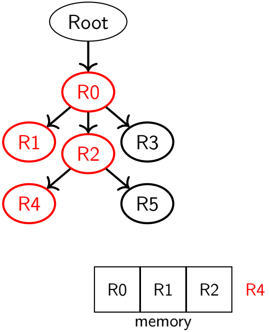
# SESE Selection Problem (SSP)

- Constraints

▶ Memory size

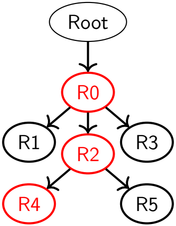
▶ Stack size

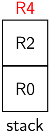
▶ Comparison per cycle



# SESE Selection Problem (SSP)

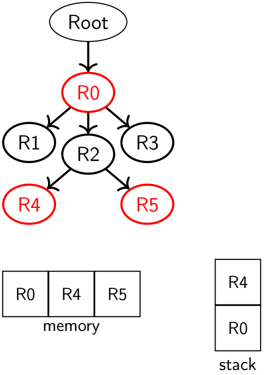
- Constraints
  - Memory size
  - Stack size
  - Comparison per cycle



# SESE Selection Problem (SSP)

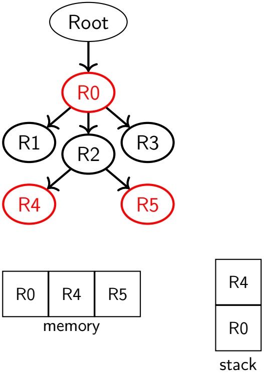
- Constraints
  - Memory size
  - Stack size
  - Comparison per cycle



# SESE Selection Problem (SSP)

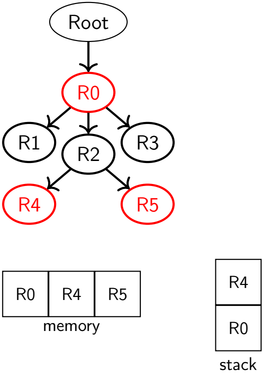
- Constraints
  - Memory size
  - Stack size
  - Comparison per cycle

- Goal
  - Cover the whole program
  - Respect constraints
  - Reduce maximal threshold



# SESE Selection Problem (SSP)

- Constraints
  - Memory size
  - Stack size
  - Comparison per cycle



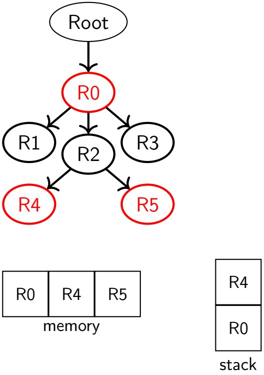

- Goal
  - Cover the whole program
  - Respect constraints
  - Reduce maximal threshold

- Maximal Threshold

$$\max_{S \in selected} (MIT\ S)$$

# SESE Selection Problem (SSP)

- Constraints
  - Memory size
  - Stack size
  - Comparison per cycle




- Goal
  - Cover the whole program
  - Respect constraints
  - Reduce maximal threshold

- Maximal Threshold

$$\max_{S \in selected} (MIT\ S)$$

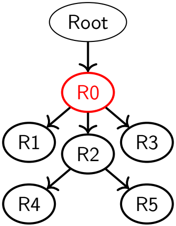
Find best *selected* set

# Solving SSP - Algorithm

Greedy algorithm : At each iteration, reduce the region with maximum threshold (= MIT)

# Solving SSP - Algorithm

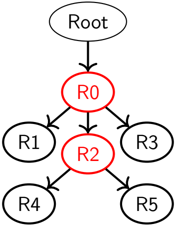
Greedy algorithm : At each iteration, reduce the region with maximum threshold (= MIT)




| region | step 0 | step 1 | step 2 |
|---|---|---|---|
| R0 | 2000 | | |
| | | | |
| | | | |

# Solving SSP - Algorithm

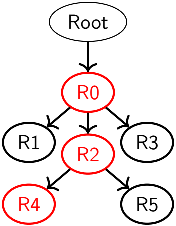
Greedy algorithm : At each iteration, reduce the region with maximum threshold (= MIT)




| region | step 0 | step 1 | step 2 |
|---|---|---|---|
| R0 | 2000 | 200 | |
| R2 | | 800 | |
| | | | |

# Solving SSP - Algorithm

Greedy algorithm : At each iteration, reduce the region with maximum threshold (= MIT)




| region | step 0 | step 1 | step 2 |
|---|---|---|---|
| R0 | 2000 | 200 | 200 |
| R2 | | 800 | 100 |
| R4 | | | 400 |

# SESE Selection algorithm properties

- Polynomial complexity

| | |
|---|---|
| MIT Estimation | $O(n^2)$ |
| Operation (without estimations) | $O(n^3)$ |

# SESE Selection algorithm properties

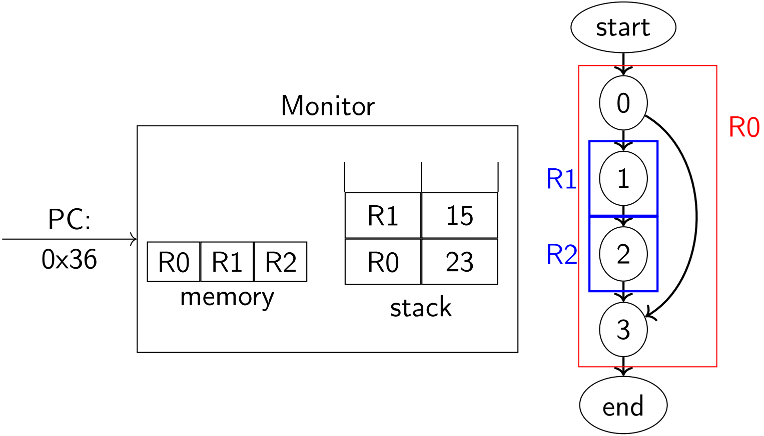
- Polynomial complexity

| | |
|---|---|
| MIT Estimation | $O(n^2)$ |
| Operation (without estimations) | $O(n^3)$ |

- Minimal threshold with infinite constraints

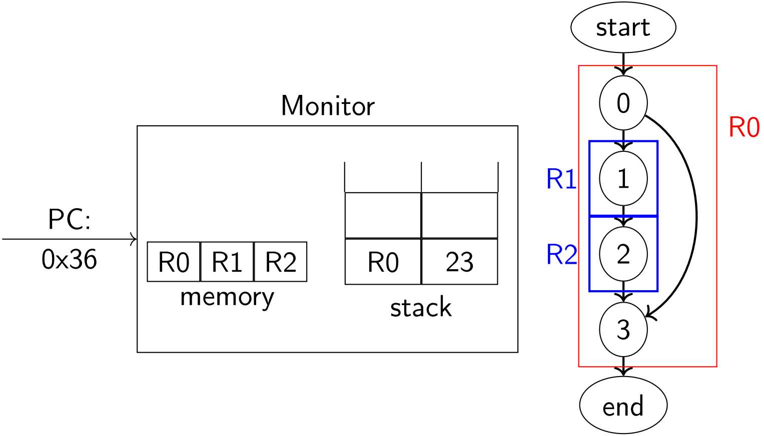
# Monitoring example

| region | entry | exit | MIT | sub-region |
|---|---|---|---|---|
| R0 | 0x0 | 0x80 | 100 | **R1, R2** |
| R1 | 0x20 | **0x36** | **25** | ∅ |
| R2 | **0x36** | 0x60 | 40 | ∅ |



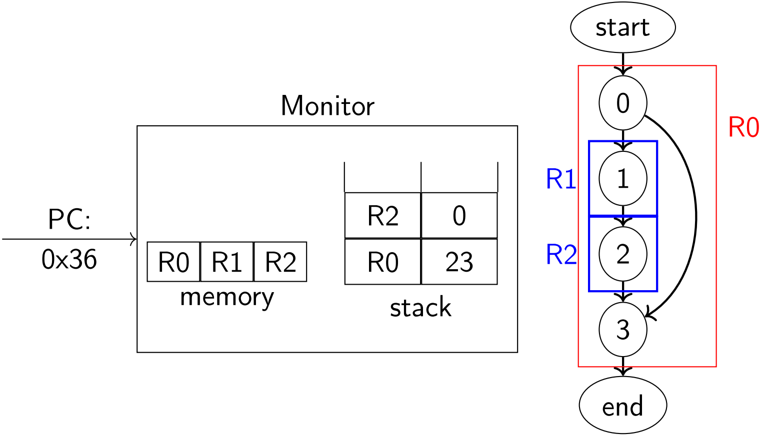
# Monitoring example

| region | entry | exit | MIT | sub-region |
|---|---|---|---|---|
| R0 | 0x0 | 0x80 | 100 | R1, R2 |
| R1 | 0x20 | **0x36** | **25** | ∅ |
| R2 | 0x36 | 0x60 | 40 | ∅ |



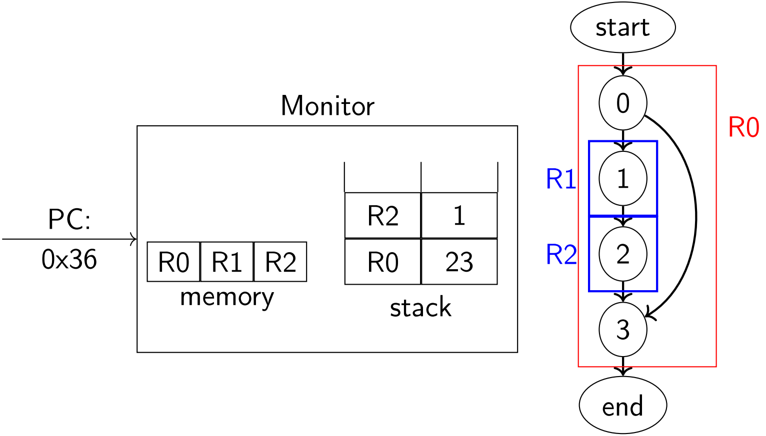
# Monitoring example

| region | entry | exit | MIT | sub-region |
|---|---|---|---|---|
| R0 | 0x0 | 0x80 | 100 | **R1, R2** |
| R1 | 0x20 | **0x36** | **25** | ∅ |
| R2 | **0x36** | 0x60 | 40 | ∅ |



# Monitoring example

| region | entry | exit | MIT | sub-region |
|---|---|---|---|---|
| R0 | 0x0 | 0x80 | 100 | **R1, R2** |
| R1 | 0x20 | **0x36** | **25** | ∅ |
| R2 | **0x36** | 0x60 | 40 | ∅ |



# Realisation

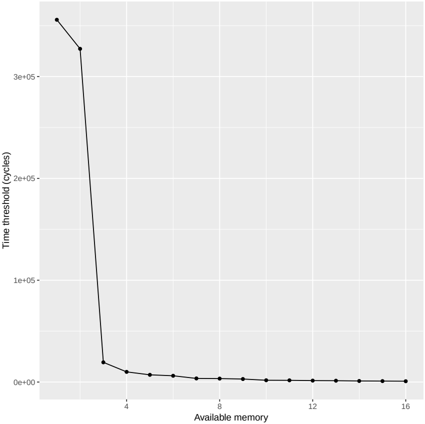
- Implemented a CFG + SESE extraction + selection algorithm for Leon3 architecture
- Implemented and tested the monitor in HLS using Catapult
- Not yet integrated on a CPU

# Experimental setup

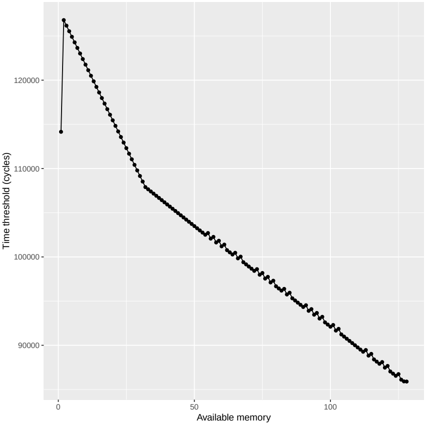
- Leon3 architecture
- Mälardalen benchmarks
- *aiT* for MIT estimation (with annotations)
- Goal : Measure the performance of the algorithm

# SESE Selection - Maximal threshold evolution

- Normal behavior

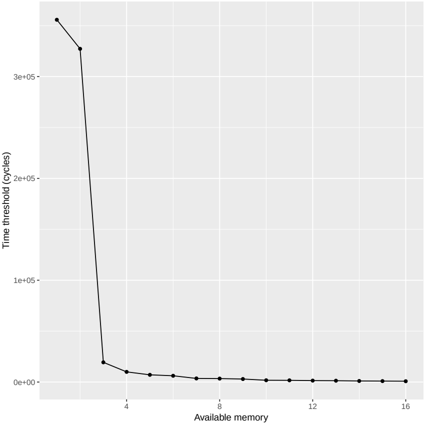



- Specific benchmark

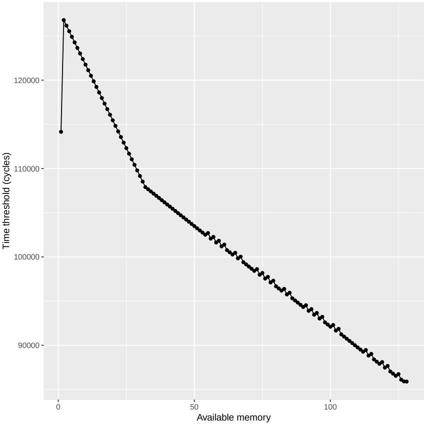


# SESE Selection - Maximal threshold evolution

- Normal behavior

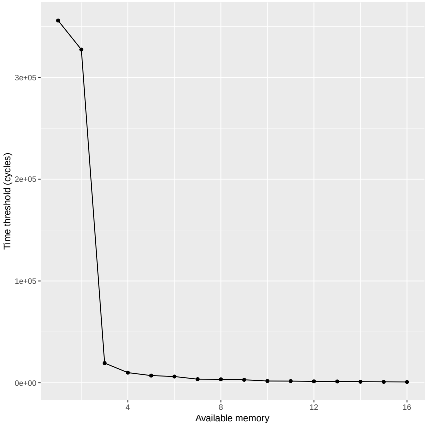



- Specific benchmark

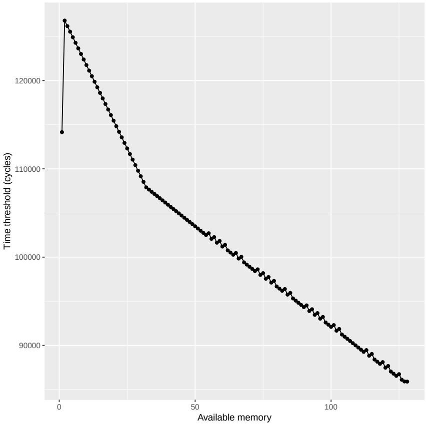


# SESE Selection - Maximal threshold evolution

- Normal behavior

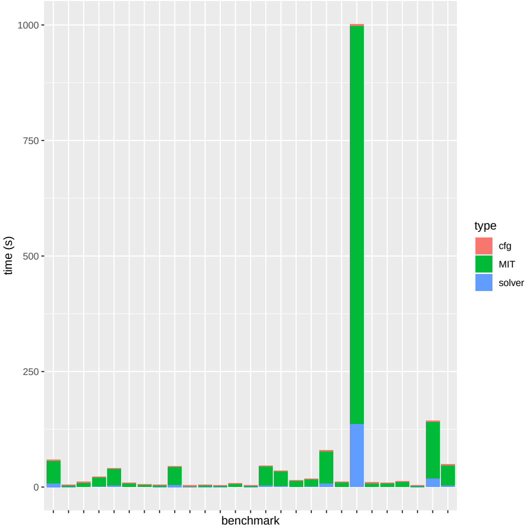



- Specific benchmark

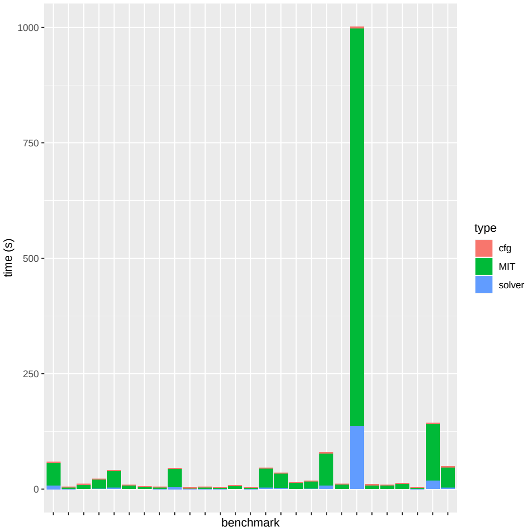



Major improvement first, small optimizations later

# SESE Selection - Runtime

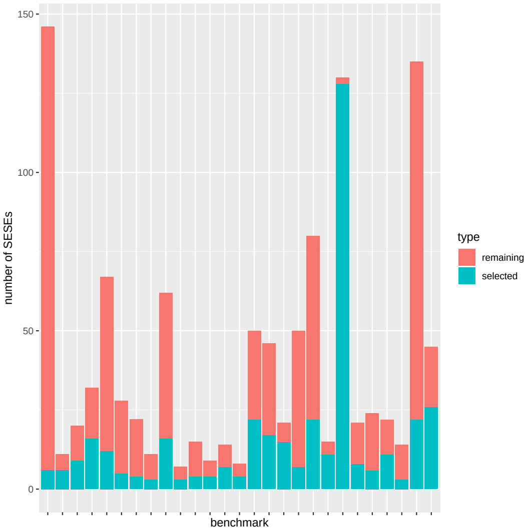


# SESE Selection - Runtime

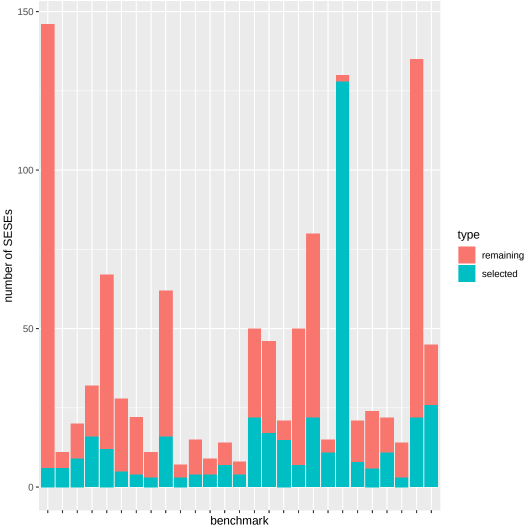



Most of the running time spent in MIT estimations

# SESE Selection - Maximal memory usage



# SESE Selection - Maximal memory usage




Minimal threshold without a time-consuming exploration

# Bibliography

[1] Reinhard Wilhelm et al. "The worst-case execution-time problem - overview of methods and survey of tools".In: ACM Trans. Embedded Comput. Syst. 2008

[2] Buttazzo, G. "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications"

[3] Kristin Krüger et al. "Vulnerability Analysis and Mitigation of Directed Timing Inference Based Attacks on Time-Triggered Systems". In: ECRTS 2018

[4] Joachim Fellmuth et al. "Instruction Caches in Static WCET Analysis of Artificially Diversified Software". In: ECRTS 2018

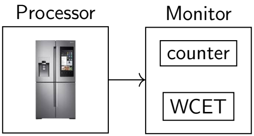
[5] Christopher Zimmer et al. "Time-based intrusion detection in cyber-physical systems". In: ICCPS 2010

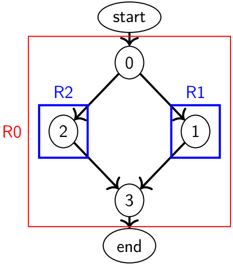
[6] Man-Ki Yoon et al. "Learning Execution Contexts from System Call Distribution for Anomaly Detection in Smart Embedded System". In: IoTDI 2017

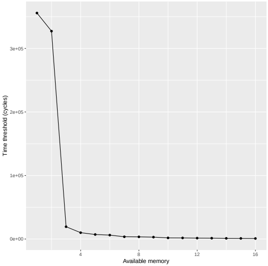
[7] Ronny Chevalier et al. "Co-processor-based Behavior Monitoring: Application to the Detection of Attacks Against the System Management Mode". In: Computer Security Applications Conference 2017

# Summary & Future Works

- Summary

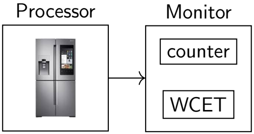


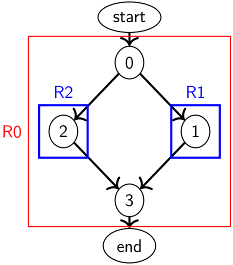



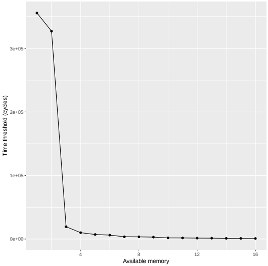




- Future Work

- Summary



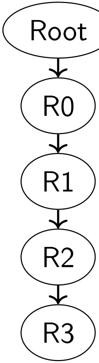

- Future Work
  - Integrate the monitor on a CPU
  - Implement fault models
  - Further reduce the threshold with inner basic block SESEs

# Solving SSP - Issues

# Solving SSP - Issues

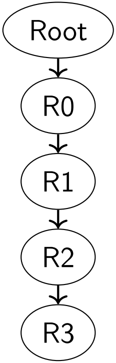
- Use Integer Linear Programing ?

# Solving SSP - Issues




- Use Integer Linear Programing ?

# Solving SSP - Issues




- Use Integer Linear Programing ?

Worst case :
$O(2^n)$ MIT estimations