KU LEUVEN

Responsiveness Guarantee for the Sancus Protected Module Architecture

Michiel Van Beirendonck

Outline

1. Motivation
2. Sancus
   - Overview
   - Secure I/O & application case
3. Objectives
   - Attacker model & properties
4. Design
   - Attack vectors
   - Extensions
5. Evaluation
   - Responsiveness argument
   - Demo
6. Conclusion & Reflection

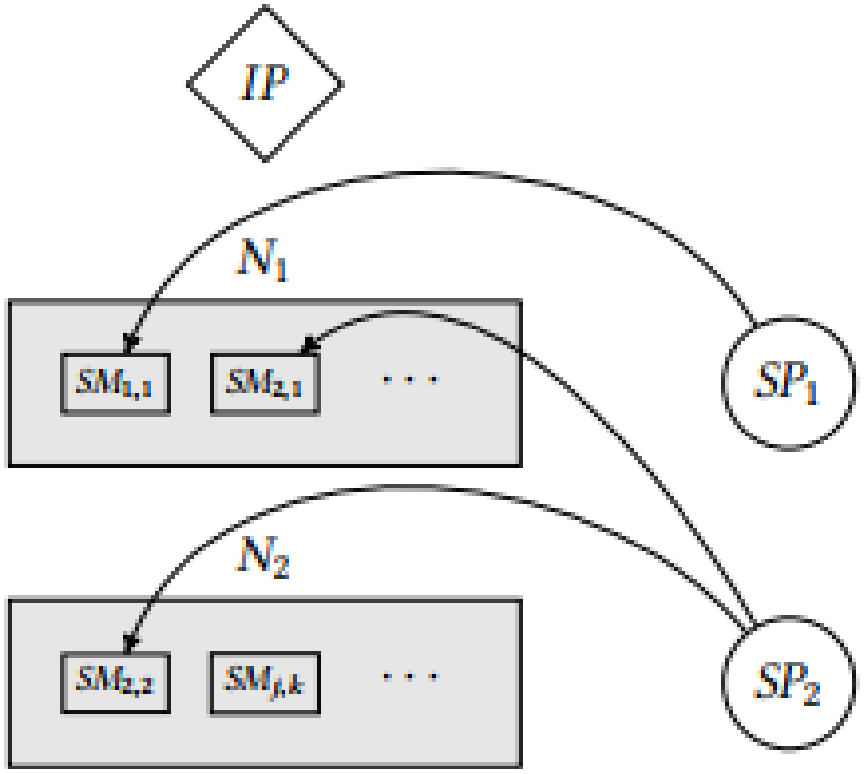
Motivation: Information Security & PMA

- Embedded systems
  - Internet connectivity
  - SW extensibility
- CIA triad
- Protected Module Architectures
  - Protected modules in shared address space
  - PCBAC

Sancus: Overview

- Low-cost
- Zero-software TCB
- Software module isolation
  - Memory access logic
- Secure communication & remote attestation
  - Key derivation:

    $K_{N,SP,SM} = kdf(K_{N,SP}, SM)$



Sancus: Secure I/O & Application Case

- Authentic Execution
  - Physical input -> application processing -> physical output
  - No responsiveness
- Application
  - Led cycle through timer interrupts

Objectives: Attacker Model & Responsiveness Properties

- Attacker
  - Controls all software
  - Controls all network communication
  - Properly implemented SM
  - No HW level attacks
- Simplification
  - After initial loading phase
  - Single trusted responsiveness domain

Protected application responsiveness based on remote attestation

  - Following ISR: response in finite interval

Design: Attack Vectors

1. Halting protected applications (Memory violations)
2. Monopolizing the CPU
3. Deploying modules
4. Overwriting crucial data structures
5. Redirecting unprotected outcalls

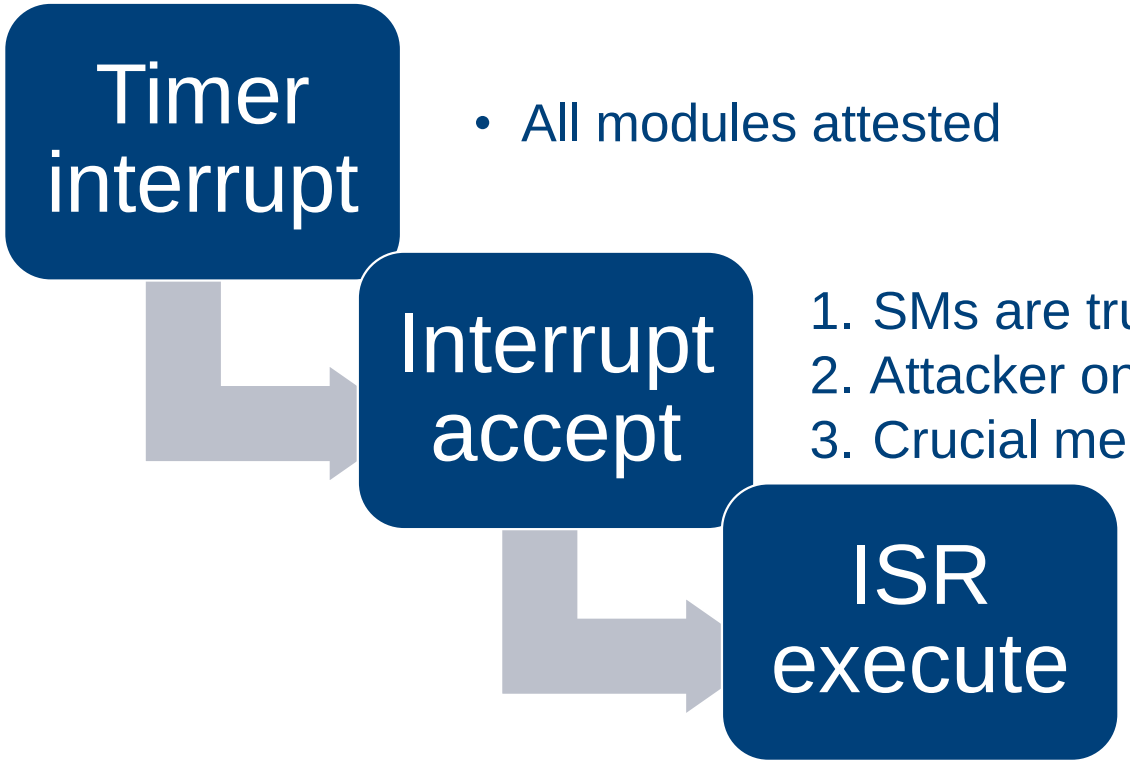
Design: Extensions

1. Halting protected applications (Memory violations)
   - Uninterruptible SM
   - Memory violation → legal action
2. Monopolizing the CPU
   - Cannot disable interrupts
   - Cannot write uninterruptible code (ISR)
3. Deploying modules
   - Exclusive access to SP_j
   - $K_{N,SP}$ to deploy more modules

Design: Extensions

4. Overwriting crucial data structures
   - Interrupt Vector Table (IVT)
     - ISR SM
   - Status Register (SR)
   - Peripherals
     - MMIO driver SM
5. Redirecting unprotected outcalls
   - Linker support to warn software developer

Evaluation: Responsiveness Argument

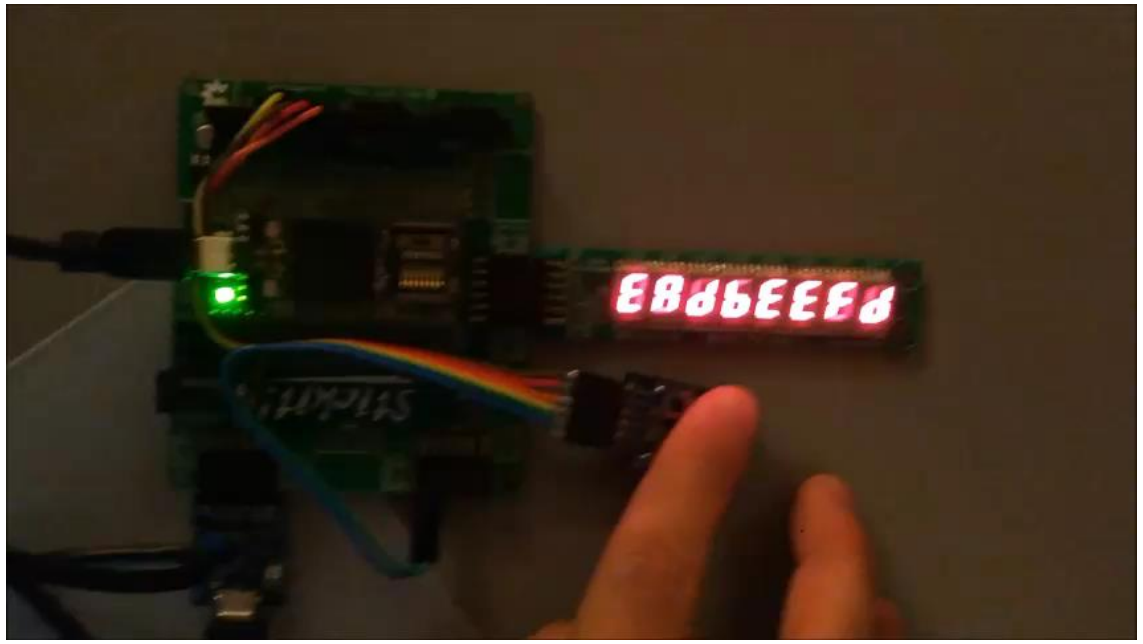


- All modules attested

1. SMs are trusted
2. Attacker only interruptible code
3. Crucial memory regions protected

1. IVT protected
2. Uninterrupible ISR

Evaluation: Demo



```
while(1)
{
  if ((P1IN & 1) == 1) //BTN1
  {
  LED1 = 0x77; // 'A'
  LED2 = 0x76; // 'H'
  LED3 = 0x77; // 'A'
  LED4 = 0x76; // 'H'
  LED5 = 0x77; // 'A'
  LED6 = 0x76; // 'H'
  LED7 = 0x77; // 'A'
  LED8 = 0x76; // 'H'
  asm("dint");
  }
}
```

Secure peripherals & Memory violation handling

Interrupt disable protected

Conclusion & Reflection

- Responsiveness guarantee relevant for safety-critical applications
- SW and HW extensions
- Evaluated in responsiveness and performance

Reflection

- Gained a lot of knowledge
- Creativity and self-criticism in scientific research
- Communication skills
- Planning

References

[1] Job Noorman, Jan Tobias Mühlberg, and Frank Piessens. 2017. Authentic Execution of Distributed Event-Driven Applications with a Small TCB. In STM ’17 (LNCS). Springer, Heidelberg. Accepted for publication.

[2] Job Noorman, Jo Van Bulck, Jan Tobias Mühlberg, Frank Piessens, Pieter Maene, Bart Preneel, Ingrid Verbauwhede, Johannes Götzfried, Tilo Müller, and Felix Freiling. 2017. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. ACM Transactions on Privacy and Security (TOPS) 20, 3 (September 2017), 7:1–7:33.

[3] Raoul Strackx, Job Noorman, Ingrid Verbauwhede, Bart Preneel, and Frank Piessens. 2013. Protected software module architectures. In ISSE 2013 Securing Electronic Business Processes. Springer, 241–251.

[4] Raoul Strackx, Frank Piessens, and Bart Preneel. 2010. Efficient isolation of trusted subsystems in embedded systems. Security and Privacy in Communication Networks (2010), 344–361.