An Open-Source Framework for Developing Heterogeneous Distributed Enclave Applications

Gianluca Scopelliti, Sepideh Pouyanrad, Jan Tobias Mühlberg

FOSDEM- Feb 2021
The distributed applications runs on huge software and hardware stacks with multiple heterogeneous vendors everywhere. Which parts are trusted?

Sensors come from heterogeneous vendors. Why would you trust them?

The cloud is “other people’s computers”. Why trust them?

Terminals may be used and managed by health care professionals. How to ensure the authenticity of data?
**TEEs: Trusted Execution Environments**

- **Isolation** of sensitive code and data
- **Authentication** of the running software (Remote Attestation)
- **Minimise Trusted Computing Base (TCB):**
  - Remove hypervisors, OSs, libraries from TCB
- **Reduction** of the attack surface
  - Only trust hardware and your own code

![Fig. 1: An Overview of TEE Building Blocks](image-url)
TEEs: Trusted Execution Environments

Intel SGX Helicopter View

- Protected enclave in application’s virtual address space
- Enclave can be entered through restrictive call gate only
- Provides attestation interface and Data Sealing
- Memory encryption defends against untrusted system software and cold boot attacks

Separating the CPU into the Normal World and the Secure World

Switching between two worlds through Monitor Mode

Memory and Peripheral Partitioning into Secure/Non-secure regions

Provides Secure Boot
Sancus: Strong and Light-Weight Embedded Security

- Extends openMSP430 with strong security primitives
  - Software Component Isolation
  - Cryptography & Attestation
  - Secure I/O through isolation of MMIO ranges

- Cryptographic key hierarchy for software attestation

- Isolated components are typically very small (< 1kLOC)

- Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/
Sancus: Strong and Light-Weight Embedded Security

- Extends openMSP430 with strong security primitives
  - Software Component Isolation
  - Cryptography & Attestation
  - Secure I/O through isolation of MMIO ranges

- Cryptographic key hierarchy for software attestation

- Isolated components are typically very small (< 1kLOC)

- Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

N = Node; SP = Software Provider / Deployer; SM = protected Software Module
Comparing Hardware-Based Trusted Computing Architectures

Adapted from “Hardware-Based Trusted Computing Architectures for Isolation and Attestation”, Maene et al., IEEE Transactions on Computers, 2017. [MGdC+ 17]
Authentic Execution

✓ Adapted from “Authentic Execution of Distributed Event-Driven Applications with a Small TCB”, Noorman et al., [STM 2017]:

"if the application produces a physical output event (e.g., turns on an LED), then there must have happened a sequence of physical input events such that that sequence, when processed by the application (as specified in the high-level source code), produces that output event."

✓ Goal: strong assurance of the secure execution of distributed event-driven applications on shared infrastructures with small TCB

✓ Its principles can be applied to any TEE -&gt; Heterogeneity!
Our Framework Features

✓ For event-driven, distributed applications

✓ Supported **heterogeneous** TEEs:
  ✓ SGX with Fortanix EDP
  ✓ Open-Source Sancus
  ✓ TrustZone with OP-TEE

✓ High Level of **Abstraction** over:
  ✓ Platform-specific TEE layer
  ✓ Secure communication API between modules

✓ Automatic deployment and Remote Attestation
A simple and secure distributed application using Sancus and SGX

FOSDEM 21
Gianluca Scopelliti

PhD student, ESR1 of the 5GhOSTS project promoted by KU Leuven and Ericsson

- “Integrity assurance for multi-component services in 5G networks”
Setup

node_sancus1

node_sancus2

de deployer

user
Setup

Button: input of the system

node_sancus1

node_sancus2

deployer

user

node_sgx
Setup

LED: toggled at every button press (ON/OFF)

node_sancus1

node_sancus2

de deployer

node_sgx

user
Setup

Store # of button presses, interface for deployer and external users
Setup

Deployment & configuration of the modules and their connections

node_sancus1

node_sancus2

Deployment & configuration of the modules and their connections

deployer

user

node_sgx
Setup

HTTP requests to get # of button presses
Deployment

node_sancus1

node_sancus2

source files, deployment descriptor

deployer

user
Deployment

- Send binaries to nodes
Deployment

› Send binaries to nodes
› Load modules
Deployment

› Send binaries to nodes
› Load modules
› Remote Attestation
   › Establishment of secure channels using module keys
Establishment of connections
Establishment of connections

node_sancus1
- led_driver
- db
- controller
- webserver
- EM

node_sancus2
- button_driver
- EM

node_sgx
- deployment descriptor
- deployer
- user
Establishment of connections

- led_driver
- button_driver
- controller
- db
- webserver
- EM
- node_sancus1
- button_pressed
- node_sancus2
- deployment descriptor
- deployer
- user
Establishment of connections
Establishment of connections
Establishment of connections

- **node_sancus1**
  - controller
  - db
  - webserver
  - EM
  - **increment** presses
  - **get_presses**
- **node_sancus2**
  - exclusive MMIO
  - button_driver
  - EM
  - **button_pressed**
  - **get_presses**

- **node_sgx**
  - led_driver
  - EM
  - toggle_led

- **deployer**
- **user**

- **KU LEUVEN**
- **DistriNet**
Establishment of connections

- **node_sancus1**
  - controller
  - db
  - webserver
  - EM
  - toggleLed
  - incrementPresses
  - getPresses

- **node_sancus2**
  - exclusive MMIO
  - button_driver
  - EM

- **deployment descriptor**

- **user**
  - HTTP GET

- **deployer**
Source code snippet: controller

```rust
/// sm_output(toggle_led)
/// sm_output(increment_presses)

/// sm_input
pub fn button_pressed(_data : &[u8]) {
    info!("Remote button has been pressed");

    // toggle LED
    toggle_led(&[]);

    // increment occurrences on db
    increment_presses(&[]);
}
```
Deployment descriptor snippet: connections

```
"connections": [
  {
    "from_module": "button_driver",
    "from_output": "button_pressed",
    "to_module": "controller",
    "to_input": "button_pressed",
    "encryption": "spongent"
  },
  {
    "name": "init-server",
    "direct": true,
    "to_module": "webserver",
    "to_input": "init",
    "encryption": "aes"
  }
]
```
Demo!

github.com/gianlu33/authentic-execution
Security discussion

Strong **integrity**

» The LED can be **only** toggled by a button press

» The value stored in the db can be **only** incremented by a button press
Security discussion

› **Strong integrity**
  » The LED can be only toggled by a button press
  » The value stored in the db can be only incremented by a button press

› **Confidentiality** of application state and sensitive data
  » TEEs
  » Secure communication channels
Security discussion

› **Strong integrity**
  » The LED can be **only** toggled by a button press
  » The value stored in the db can be **only** incremented by a button press

› **Confidentiality** of application state and sensitive data
  » TEEs
  » Secure communication channels

› **Availability out of scope**
  » nothing happens if, e.g., an event is lost
Future work

- TrustZone support - available soon
  - with Rust

- More flexible deployment tools
  - stop/migrate a module
  - deploy new modules after first deployment

- SGX improvements
  - Remote Attestation using Fortanix CCM
  - performance
  - sealing
Thank you!