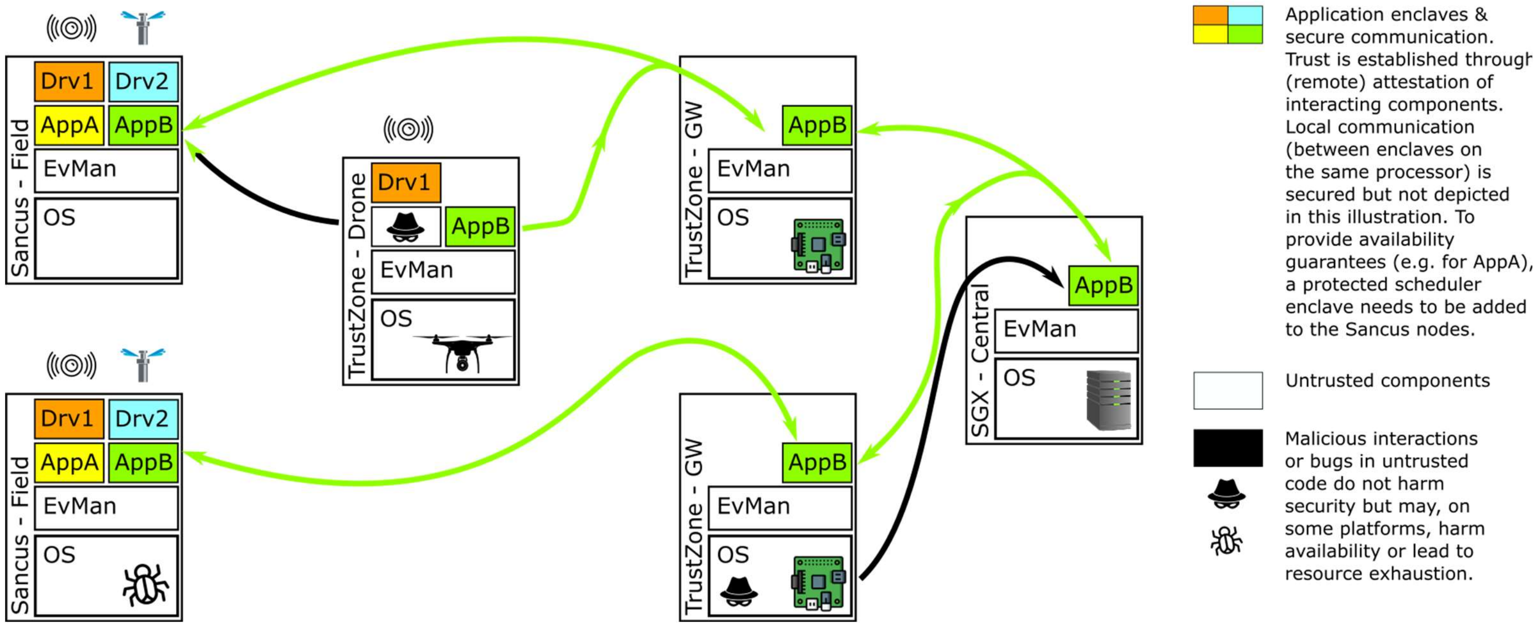
DistriNet KU LEUVEN

An Open-Source Framework for Developing Heterogeneous Distributed Enclave Applications

Gianluca Scopelliti, Sepideh Pouyanrad, Job Noorman,
Fritz Alder, Frank Piessens, Jan Tobias Mühlberg

The 28th ACM Conference on Computer and Communications Security
ACM CCS 2021



Motivations

Security for emerging safety-critical use cases in precision agriculture, smart energy systems, smart mobility

- Strong authenticity guarantees in heterogeneous open Internet-of-Things and Cyber-Physical Systems with Edge and Cloud
- Leverage different Trusted Execution Environments (TEEs), easing development effort, reactive/event-driven development model
- Focus on dependable safety-critical systems with sensing/actuation

Security properties

Authentic Execution / Robust Safety

- Physical system outputs can be explained in terms of physical inputs and application source code (assuming correct compilation and no bugs or vulnerabilities in the application enclaves)
- Secure I/O: Only attested application components can access I/O devices; attackers cannot interfere with I/O at software level
- Additionally provides limited confidentiality and availability

Our framework


```c
// declare output using a macro
SM_OUTPUT(sensor_driver, sensor_read);

void read_from_sensor(void) {
  int sensor_value = read_from_io_device();

  // output call -> generate new event
  sensor_read(&sensor_value, 2);
}
```

```yaml
# deployment descriptor

connections:
-
  from_module: sensor_driver
  from_output: sensor_read
  to_module  : aggregator
  to_input   : receive_data
```

```rust
// declare input using an annotation
//@ sm_input
pub fn receive_data(data : &[u8]) {
  let sensor_value = parse_data(data);

  // do something with data
}
```

End-to-end security

- Supported TEEs: Intel SGX, ARM TrustZone (with OP-TEE), Sancus
- Automated deployment, attestation, key management
- Authenticated Encryption to protect communication channels
- Secure I/O provided by Sancus

Reduced development effort

- Simple event-driven programming model
- *Declarative* approach: code annotations and deployment descriptor
- Automatic enclaved execution and attestation
- Automatic establishment of secure channels

Preliminary results

Development effort

button-led example: *github.com/AuthenticExecution/examples*

- 7 to 123 LOC for developing SGX or Sancus modules
- TrustZone: ~1kLOC of which only 58 LOC are app logic
- Deployment descriptor: 137 LOC

Round-Trip Time (RTT) SGX-TrustZone-Sancus

Avg. RTT for 8 bytes of (encrypted) payload: *256.22 ms* (!)

- TZ's implementation of SPONGENT (Sancus' crypto engine): 160.5 ms
- TZ's slow transition NW<->SW: 18.22 ms
- Other issues: TZ emulation (QEMU), slow networking (UART, SLIRP)