POSTER: An Open-Source Framework for Developing Heterogeneous Distributed Enclave Applications

Gianluca Scopelliti
gianluca.scopelliti@ericsson.com
imec-DistriNet
KU Leuven
Leuven, Belgium

Sepideh Pouyanrad
sepideh.pouyanrad@kuleuven.be
imec-DistriNet
KU Leuven
Leuven, Belgium

Job Noorman
job.noorman@kuleuven.be
imec-DistriNet
KU Leuven
Leuven, Belgium

Fritz Alder
fritz.alder@acm.org
imec-DistriNet
KU Leuven
Leuven, Belgium

Frank Piessens
frank.piessens@kuleuven.be
imec-DistriNet
KU Leuven
Leuven, Belgium

Jan Tobias Mühlberg
jantobias.muehlberg@kuleuven.be
imec-DistriNet
KU Leuven
Leuven, Belgium

ABSTRACT

We present an integrated open-source framework to develop, deploy, and use event-driven distributed enclaved applications across heterogeneous Trusted Execution Environments (TEEs). Our framework strives for strong application authenticity and integrity guarantees, and optionally confidentiality and availability, while minimizing the Trusted Computing Base (TCB). For software developers, our framework provides a high level of abstraction over the platform-specific TEE layer that provides isolation, attestation and secure communication amongst distributed application components, allowing developers to focus of application logic. We provide a notion of event-driven programming to develop distributed enclave applications in Rust and C for heterogeneous TEEs, including Intel SGX, ARM TrustZone and the open-source Sancus. This heterogeneity makes our framework uniquely suitable for a broad range of use cases which combine cloud processing, mobile and edge devices, and lightweight sensing and actuation.

CCS CONCEPTS

• **Security and privacy** → **Trusted computing**; **Distributed systems security**; • **Computer systems organization** → **Sensors and actuators**; *Availability*; *Maintainability and maintenance.*

KEYWORDS

Trusted Execution; Event-Driven Systems; Intel SGX; ARM TrustZone; Sancus

ACM Reference Format:
Gianluca Scopelliti, Sepideh Pouyanrad, Job Noorman, Fritz Alder, Frank Piessens, and Jan Tobias Mühlberg. 2021. POSTER: An Open-Source Framework for Developing Heterogeneous Distributed Enclave Applications. In *Proceedings of ACM Conference on Computer and Communications Security (Submission to ACM CCS 2021).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Submission to ACM CCS 2021, Due 16 August 2021, Coex, Seoul, South Korea
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION & PROBLEM

Trusted Execution Environments (TEEs) allow an application to execute in a hardware-protected environment called *enclave*. Enclaves are isolated and protected from the rest of the system, ensuring strong confidentiality and integrity guarantees. Cryptographic primitives and cryptographic keys, which are unique per enclave and which can only be used by that enclave, enable secure communication and remote attestation; the latter is a mechanism to obtain cryptographic proof that an application is running under enclave protection on a specific processor. There are several TEEs available, both in industry and research. Open-source TEEs include Sancus and Keystone; proprietary options are, e.g., SGX for Intel processors, SEV for AMD, TrustZone for ARM, and others [3]. Developing distributed applications that execute on heterogeneous TEEs is difficult, in particular for scenarios that combine Internet-of-Things, Edge, and cloud hardware: each TEE requires a platform-specific software implementation, comes with different approaches to key management and attestation, a different Trusted Computing Base (TCB) footprint, and provides slightly different hardware features and security guarantees.

Therefore, developing a distributed application that uses a multitude of TEEs architectures is non-trivial. A developer needs to make choices as to which security features are required for which components, adapt the code of each component to multiple specific platforms, arrange for different deployment and attestation strategies, and implement secure interaction between the components. Open-source projects such as Open Enclave SDK and Google Asylo aim to bridge the development gap between different TEEs. However, software engineers still need to account for the communication between different modules, which has to be properly secured with cryptographic operations for data encryption and authentication. In particular, the responsibility for deploying the distributed application, loading and attesting each enclave, establishing session keys and secure connections between distributed components, is still left to the application developer and operator. Overall, ensuring strong security guarantees in distributed scenarios poses a challenge to the adoption of TEE technology. To address these challenges, our framework makes the following contributions:

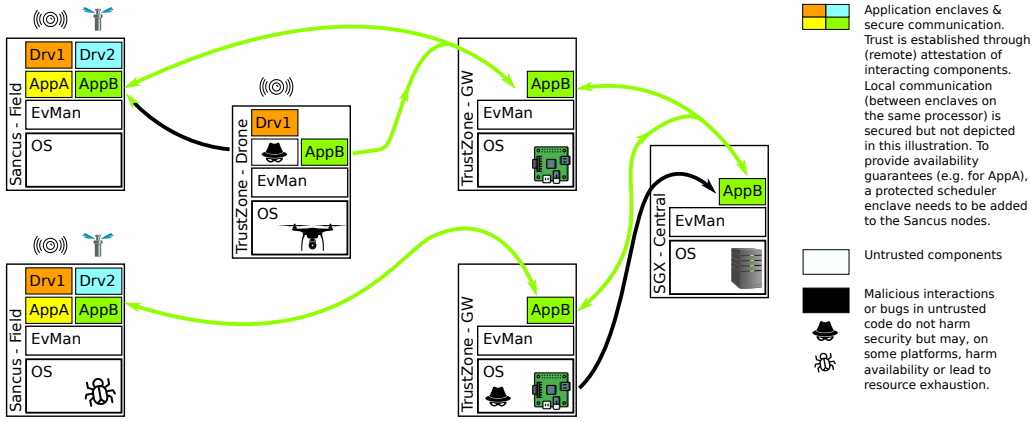

Figure 1: A smart irrigation system as an example for distributed application networks we support. Light-weight sensing and actuation nodes are deployed in a field. Application *AppA* controls irrigation units (through driver *Drv2*) based on soil moisture (obtained through *Drv1*). Application *AppB* provides the same functionality but has access to additional data sources, e.g., aerial surveillance and data aggregation on central infrastructure. All application components execute in enclaves (colored). Directed data flows through untrusted networks (colored arrows) are at least authenticated and integrity protected; attestation precedes the establishment of all data flows, and a notion of local attestation is used to establish trust between enclaves on the same processor. All other software in the scenario is untrusted regarding our security properties, which leads to a very small run-time application TCB. Guaranteeing availability properties may require a different compartmentalization strategy. The concept is also applicable across, e.g., the different control units within a car or in an autonomous robot.

- We present an integrated approach for the authentic execution of event-driven programs on heterogeneous distributed systems, under the assumption that the execution infrastructure offers specific security primitives – TEEs with support for secure I/O (cf. [6]) and real-time processing (cf. [2]);
- We integrate a technique for implementing support for secure I/O by means of protected driver modules, and availability through TEE extensions, on small TEE-microprocessors such as Sancus [6];
- We provide a revised open-source implementation of the approach for Intel SGX, ARM TrustZone, and for Sancus, which supports software development in Rust and C;
- We work towards an extensive evaluation of performance and security aspects of that implementation. Preliminary results show that our framework allows for the deployment of complex distributed software systems with a very small run-time application TCB.

Our framework is available under an open-source license at https://github.com/AuthenticExecution/env.

2 AUTHENTIC EXECUTION

We developed the concept of *authentic execution* [5] to address the problem of securely executing distributed applications on a shared infrastructure and to also minimize the application's runtime TCB. Authentic execution provides a notion of security that we summarize as "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," which is roughly equivalent to the concept of *robust safety* in later literature [1]. This guarantee relies on standard TEE security properties – i.e., strong software isolation and software attestation – but also on a notion of *secure I/O* where physical I/O channels can be connected to an enclave such that the application enclaves maintain exclusive access over I/O peripherals.

Initially, our approach did not consider confidentiality and availability. Yet, our TEE-design [6] and a series of case studies in application domains such as smart electricity metering, smart agriculture and secure vehicular communications [4, 7, 8] did consider and provide confidentiality. Most recently, in [2], we address the availability aspect and extend light-weight embedded TEE architectures so as to provide strong temporal isolation guarantees to multiple, mutually distrusting applications. To discuss the interplay of these different concepts, we introduce an irrigation system as a use case.

2.1 Use Case: Automated Irrigation

Smart farming applications are an essential part of modern critical infrastructure. An automated irrigation system, as illustrated in Figure 1, would involve a series of light-weight sensors and actuators in the field that monitor soil moisture and control water supply . The system can be connected to edge infrastructure or cloud services for centralized configuration and maintenance, to integrate reporting and billing, and to minimize water consumption based on weather predictions. Naturally, smart farming systems are security critical since malicious interactions can potentially lead to huge costs and may destroy a crop; they also demand a high level of dependability where events must be guaranteed to be processed in a timely manner. With our approach to building such applications as event-driven systems, we intend to support developers with an intuitive programming paradigm and automated enclave deployment to achieve the security objectives highlighted below.

2.2 Security Objective

We consider an *open system* as the basis for our framework. In this open system, software is deployed dynamically and multiple stakeholders may run applications on the same infrastructure, including on the light-weight IoT and Edge hardware. Thus, we consider scenarios where arbitrary new code can be loaded at at run time and we consider powerful attackers that can manipulate all the software on the infrastructure (unless that software is isolated in enclaves), can manipulate network traffic, but cannot break crypto. Attacks against the hardware are out of scope.

Under this attacker model, isolation and mutual attestation of all application components, including peripheral driver, results in the initial authenticity and integrity guarantees, where all observed outputs can be explained in terms of a trace of (authentic) inputs and the (integrity protected) source code of the application. Using the sealing capabilities of TEEs, this guarantee can be extended to also provide confidentiality of application events and state. Our availability extensions allow individual enclaves to execute with strong guarantees for responsiveness and progress, which allows for these enclaves to timely detect and react upon availability issues across the application. We currently do not consider protection from side-channel leakage as part of the framework but as part of the developer's task to address.

2.3 Secure I/O

Several TEEs such as ARM TrustZone and Sancus allow for I/O peripherals to be exclusively controlled by the secure world or by an enclave. In the case of Sancus, e.g., peripherals are controlled through Memory-Mapped I/O. By mapping enclave memory over a peripheral's memory addresses, a driver enclave can gain exclusive control and manage access to the peripheral. In our framework, these driver enclaves effectively translate physical input and output channels into application events and vice versa. This notion of secure I/O is essential for our security guarantee as it prevents application inputs or outputs to be spoofed by software that is not part of the attested application TCB.

2.4 Availability Guarantees

In [2], we extend Sancus towards a configurable security architecture that provides a notion of guaranteed real-time execution for dynamically loaded enclaves. We implement preemptive multitasking and restricted atomicity on top of strong software isolation and software attestation. Our approach enables the hardware to enforce confidentiality and integrity protections while a decoupled small software component can enforce availability and guarantee strict deadlines of a bounded number of protected applications, without introducing a notion of priorities amongst these applications. This allows us to develop enclaves that can handle interrupts and make progress with deterministic activation latencies, even in the presence of a strong adversary with arbitrary code execution capabilities. In the context of our authentic execution framework, these enclaves can serve as a means to implement dependable sensing and control loops, but also to provide a reliable notion of time and progress across a distributed application, and to detect the lack of progress in such applications.

2.5 Software Architecture & Deployment

Application code is developed using macros and annotations to define the name and scope of enclaves, and to label input and output channels of each application component. A *deployment descriptor* specifies which component is to be loaded on which target machine, and how the respective input and output channels are to be linked together, and what communications interfaces are being used. Component loading and communication are facilitated by infrastructure software, which is untrusted regarding our security properties but needs to be trusted regarding availability. Our framework provides an enclave that facilitates the initial attestation and key management steps during the automated deployment phase. We build upon established software development tool chains for the respective source languages and target architectures.

3 SUMMARY

We presented an open-source framework for developing heterogeneous distributed enclave applications with an event-driven programming model. Our approach is distinguished by enabling an open-system approach where distrusting stakeholders can share processing resources, by supporting a range of TEEs, from the cloud to edge and IoT devices, and by providing a unique set of security and availability guarantees that enable advanced use cases, e.g., in safety-critical sensing and actuation.

ACKNOWLEDGMENTS

This research is partially funded by the Research Fund KU Leuven and by the Flemish Research Programme Cybersecurity. Specific funding was provided under the SAFETEE project by the Research Fund KU Leuven. This research has received funding under EU H2020 MSCA-ITN action 5GhOSTS, grant agreement no. 814035. Fritz Alder is supported by the Research Foundation Flanders.

REFERENCES

[1] Carmine Abate, Roberto Blanco, Deepak Garg, Catalin Hritcu, Marco Patrignani, and Jérémy Thibault. 2019. Journey beyond full abstraction: Exploring robust property preservation for secure compilation. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE, 256–25615.

[2] Fritz Alder, Jo Van Bulck, Frank Piessens, and Jan Tobias Mühlberg. 2021. Aion: Enabling Open Systems through Strong Availability Guarantees for Enclaves. In *CCS '21*. ACM, Seoul, South Korea.

[3] P. Maene, J. Götzfried, R. de Clercq, T. Müller, F. Freiling, and I. Verbauwhede. 2017. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Trans. Comput.* 99 (2017).

[4] Jan Tobias Mühlberg, Sara Cleemput, A. Mustafa Mustafa, Jo Van Bulck, Bart Preneel, and Frank Piessens. 2016. An Implementation of a High Assurance Smart Meter using Protected Module Architectures. In *WISTP '16 (LNCS, Vol. 9895)*. Springer, Heidelberg, 53–69.

[5] 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.

[6] 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 (2017), 7:1–7:33. Issue 3. https://www.beetzsee.de/posts/papers/2017-tops-sancus2.pdf

[7] Gianluca Scopelliti. 2020. *Securing Smart Environments with Authentic Execution*. Master's thesis. Politecnico Di Torino. https://distrinet.cs.kuleuven.be/software/sancus/publications/scopelliti2020.pdf.

[8] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. 2017. VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks. In *ACSAC '17*. ACM, New York, NY, USA, 225–237.