Attested Trusted Execution Environments for Transactional Workloads

I. Problem

Outsourcing all kinds of data processing to external cloud infrastructures has become best practice. A hosted computing provider may guarantee a certain amount of liability to encourage users to run their software in the provider's computing environment. IT security threats and vulnerabilities are rising exponentially. By offloading sensitive workloads to trustworthy computing environments, users may counter these threats. However, existing TEE technologies fail to address many rudimentary requirements such as ease-of-use, isolation, attestation, management, and scalability.

From our experience in finance and government, *Trusted Execution Environments* should provide at least the following properties:

- Physically and logically protect the execution of the application code from any interference and leakage (Isolation Property).
- Attest to a verifier the runtime integrity of an application, including the whole TCB.
- Creating an immutable record of the execution of the application code, detailing input, output, time, and device state (Audit Property).
- Restrict access to start an execution to authorized entities (Authorization Property).

A TEE should prove that a certain output was generated from a specific input, executed at a particular time with specific code.

II. Threat Model

Gapfruit TEE aims to provide a toolkit for hardware manufacturers and system integrators to build solutions for a variety of threat models and use-cases.

For this discussion, our threat boundary is the TEE appliance. Mitigations for hardware-based attacks are essential to get right. That's why we work closely together with hardware manufacturers, such as HSM vendors, to form a hardware/software co-design. Since every hardware manufacturer has different techniques, we consider hardware-based attacks out of scope for this analysis.

We want to guarantee the protection of integrity, confidentiality, and availability of the following assets against adversaries listed in Table I: Application, including its complete TCB, during runtime, its input, and its output.

Table I
Adversary for Asset - TEE Appliance

Adversary	Example
Network attacker	The attacker may connect to the system by network in order to eavesdrop, intercept, or modify the network packets.
	The attacker may connect to the system by network in order to exploit a vulnerability in the network stack or gRPC server.
	The attacker may connect to the system and send unauthorized *ExecuteRequest*.
Unprivileged software attacker	The attacker may hide malicious code within a TEE application that tries to escape the isolation and attack the platform or TEE applications from other security domains.
	The attacker may send input as part of an *ExecuteRequest* that triggers a vulnerability within the TEE application.
Privileged admin attacker	The attacker may try to load a TEE application into their security domain that tries to escape the isolation and attack the platform or TEE applications from other security domains.

III. Solution

We present a generic TEE toolkit applicable for many use-cases in finance, healthcare, or government. Gapfruit TEE embodies a microkernel operating system with capability-based security. Each component of the system is strongly isolated and has only access to the resources and services it really needs. The trust graph of each component is concisely defined and verified during build, deployment, and run time. Gapfruit TEE runs on ARM, x86, and RISC-V leveraging the hardware security properties these chips provide. The capability-based architecture of Gapfruit allows governing the trust relationship of each sub-system down to the hardware.

For device and appliance manufacturers, Gapfruit TEE can be customized and integrated into their products. For IT or OT system architects, products running Gapfruit TEE, are easily integrated to their infrastructure, with an open and straightforward API.

Existing applications run inside Gapfruit TEEs without or with minimal modifications. The interface to exchange data to TEE applications is via the simple Unix interfaces: *stdin*, *stdout,* and *stderr*. This allows developers to write and test applications for Gapfruit TEE on any operating system. These applications can then be deployed to a TEE appliance. The API makes it easy to handle access control and have an attested proof of the computation, the input, output, and the applications TCB during runtime.

The Gapfruit TEE provides confidence in the absence of functional impurity, while the TCB's full transitive closure is measured.

Gapfruit TEE is currently in use in the finance, health, and governmental sector as part of Securosys Imunes [1].

References

[1] Securosys Imunes
https://www.gapfruit.com/case-studies