Towards Integrating Trusted Execution Environment into Autonomic Systems

Mohamed Sabt[1,2], Mohammed Achemlal[1,3], Abdelmadjid Bouabdallah[2]

[1] Orange Labs, Caen; [2] Sorbonne Universités, UTC; [3] Greyc, EnsiCaen

ICAC 2015 — July 9, 2015

Problem

- **Self-protection** and **self-healing** are two of the main properties of autonomic computing;
- The **module** assuring these two properties must be **reliable**, that is it shall protect itself from **any malicious attacks**;
- The question is: how the **trustworthiness** of such module can be **guaranteed** ?

Background

- **Trusted Execution environment (TEE)** is a tamper-resistant processing environment adopting the dual-execution environment approach;
- It guarantees the **authenticity** of the executed code, the **integrity** of the runtime states, and the **confidentiality** of its data stored on a persistent memory;
- Very often, TEE is based on **ARM TrustZone**, a popular set of security extensions allowing asymmetric virtualization of two cores with minimum overhead;

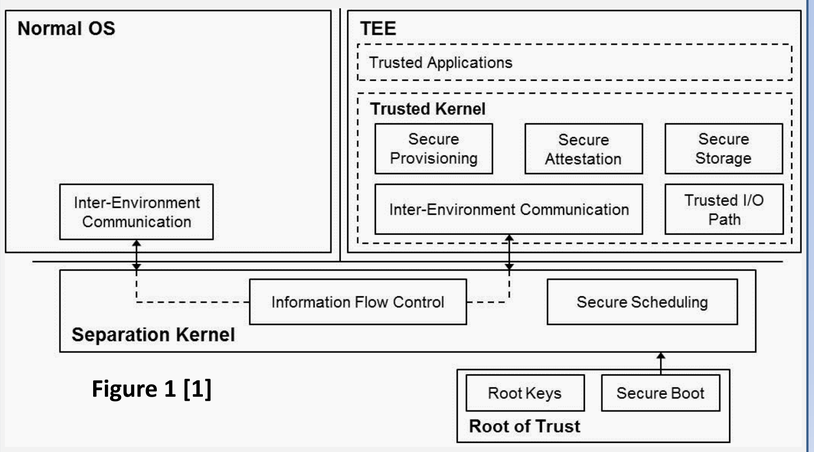

Figure 1 [1]

- TEE is **everywhere**, it might already be inside your smartphone. To verify, just install the Android app "**TEE Checker**".

Architecture

- **Autonomic manager** [2] guarantees self-protection by running a permanent loop of ***monitoring-analysis-deciding-execution***;
- It must be shielded inside a **tamper-resistant environment** in order to guarantee its proper functioning;

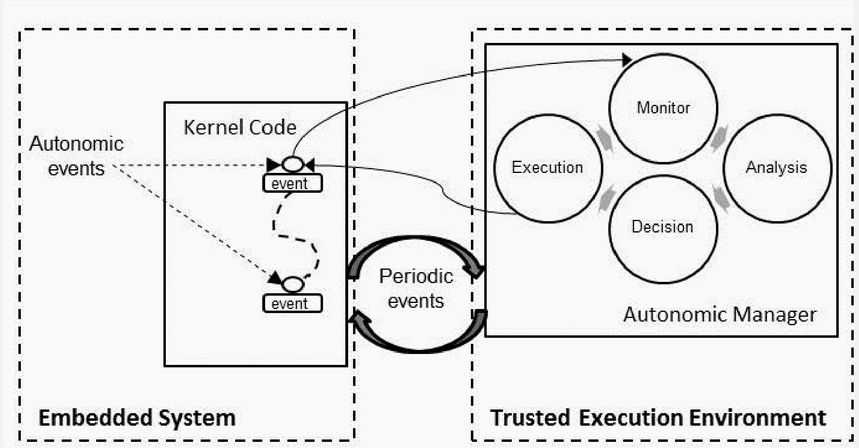

Figure 2

The security mechanisms of the autonomic manager are triggered by two types of exceptions:

1. Software Interrupt Exceptions (autonomic events):
Triggered before the execution of certain privileged instructions which the kernel is deprived from;

2. Periodic Exceptions:
Triggered by a secure timer isolated inside the secure world of TrustZone.

The TEE guarantees that these exceptions are **non-bypassable** and their handlers are **strongly isolated** from the rest of the system.

Security Mechanisms

Trap and emulate (self-protection)

- The execution of critical instructions, such as memory management, inside the kernel triggers autonomic events;
- The autonomic manager running inside the TEE executes these instructions after a thorough inspection;
- For instance, malicious attacks cannot inject code to the kernel because any write instruction to the memory must be approved before execution.

Introspection (self-healing)

- The integrity state of the loaded kernel is constantly monitored after each periodic event;
- The autonomic manager forces the system, for instance, to restart or even to halt running if it detects the non-integrity of the running kernel code.

Conclusion and Perspectives

- This work aims to use some of the trusted computing concepts to design more **trustworthy autonomic systems**;
- **Future work** will focus on precisely defining the security mechanisms running inside the autonomic manager shielded by the TEE;
- TrustZone will be used in order to significantly **reduce the overhead** incurred by the execution of these mechanisms.

Bibliography

[1] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is and what it is not," *2015 IEEE 14th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom),* in press.

[2] F. Duan, X. Li, Y. Liu, and Y. Fang, "Towards Autonomic Computing: A New Self-Management Method," in *Artificial Intelligence and Computational Intelligence, ser. Lecture Notes in Computer Science*, 2011, vol. 7002, pp. 292–299.