

Improving TDX Linux integration: changes in TDX Migration and Attestation

LPC 2026, Confidential Computing MC

Elena Reshetova, Intel

Outline

- Service TD concept and its problems for Linux
- Approach change: TDX Module extensions
- Usecases: TPA, Quoting Service, Migration Setup, vTPM
- Scheduling caveats for KVM/QEMU
- Lessons learned so far

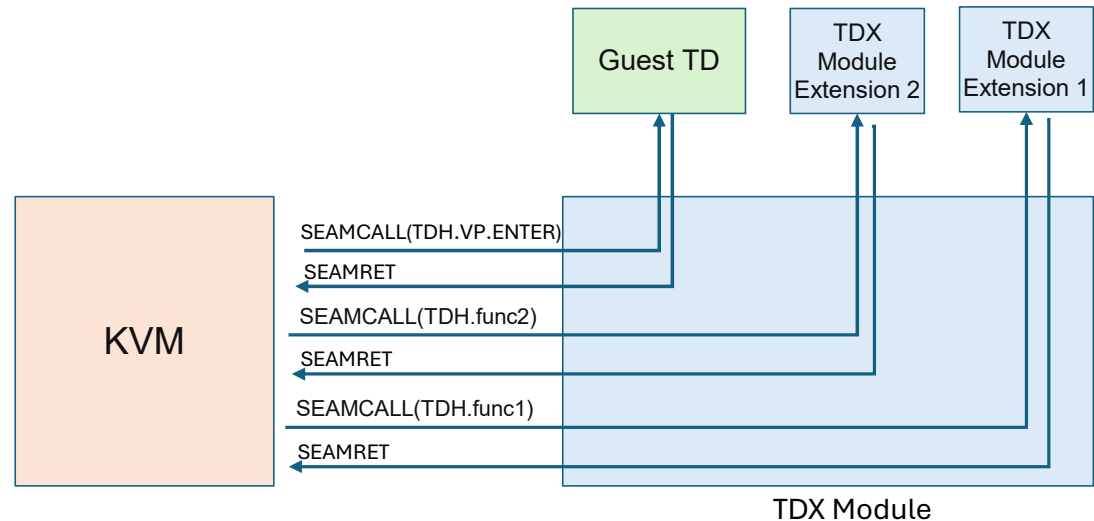
Service TD concept and its problems

- Original problem statement:
 - Complex and costly to implement arbitrary long running flows inside TDX Module
 - Requires periodic polling for interrupts, saving execution state on host exits, etc
 - Breaking long running flows into a set of subsequent SEAMCALLs requires lots of security validation and checks
 - Input checking, Order of invocation, preservation of parameters ...
- Service TDs are system TDs created to solve this problem
 - Naturally interruptible
 - Existing mechanism to save/restore VCPU state
 - Low complexity inside TDX Module
- Requires explicit host kernel/VMM management during its lifecycle
 - Create, allocate all resources (VCPU, memory), execute when required, destroy and free the resources
 - Requires VMM to bind and pre-bind such TDs to user TDs, re-bind during upgrades
 - This is costly in terms of additional kvm/qemu code complexity and overall orchestration layer on CSP side

Approach change: TDX Module extensions

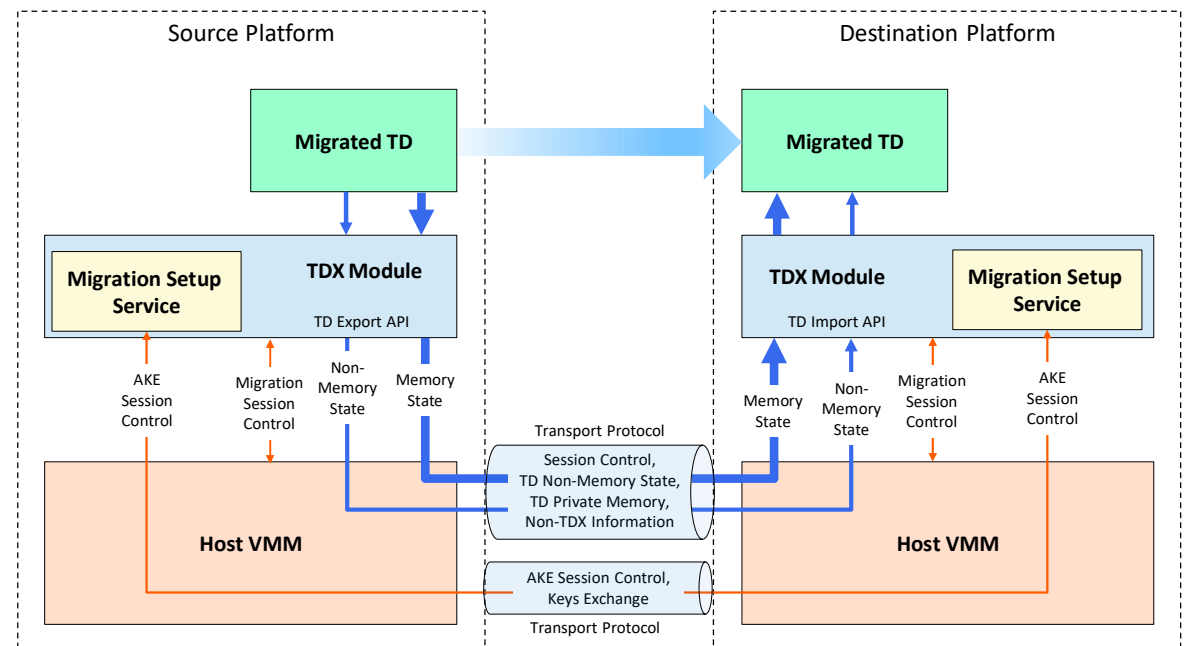
A good balance of complexity between TDX module and host kernel/VMM:

- Almost invisible to host/VMM
 - Initial initialization and additional memory provisioning to TDX Module
 - Minimal runtime impact
 - SEAMCALLs backed up by extensions look like normal interruptible SEAMCALLs
 - Caveat: Concurrency requirements
- Currently implemented as hidden TDs which brings back most of the Service TD benefits for TDX Module
 - Interruptability, vcpu state management, etc.



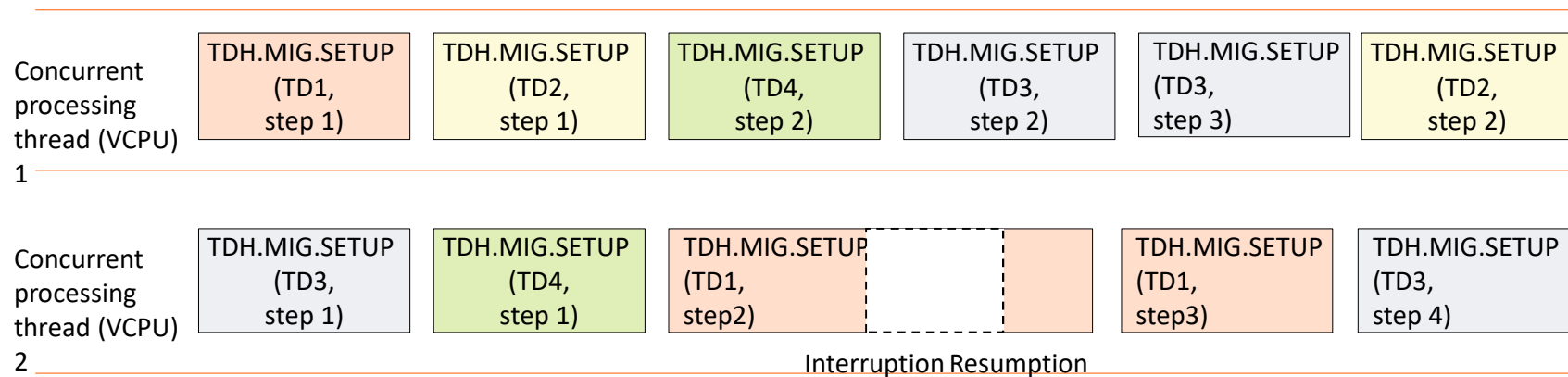
Use cases

- TEE-IO Provisioning Agent (TPA) for SPDm connection management
- Quoting Service for DICE-based TDX Attestation
- Migration Setup Service for TDX Migration establishment
- Ephemeral vTPM (potentially)
- ...



Concurrency requirements

- Each TDX Module extension can be configured to use X # of concurrent processing threads
 - Fixed configuration at TDX module initialization
- An interrupted request must be resumed on the same thread.
 - No other request can be called for the same thread until the interrupted request is called and completed
- This must be considered by qemu/KVM when issuing corresponding TDH.* SEAMCALL



Lessons learned so far

- Expose primitives and state, but Linux should decide how resources are managed
 - Only include policies inside coco architecture if required for security
- Preserve existing Linux abstractions
 - If coco requires exceptions, unify between coco architectures
- Balancing Security Against Operational Simplicity
 - Complexity might not be possible to remove without hurting use cases
 - Complexity can be moved/balanced between components
- Explain well design decisions, threat model, assumptions and restrictions
 - Hard to precisely find the right format, especially for big, complex and having legacy baggage features
- Linux code minimization, unification of data structures, coding patterns

Summary/Call to Action

- Separate Service TDs were road blocking enablement of key TDX features
- Embraced the need for an architecture that delivered TDX module semantics to address OS concerns
- Addresses numerous use-cases and it is work in progress
- Feedback is warmly welcomed
 - TDX extensions and its design principles
 - new usecases that can be enabled
 - overall principles to make Linux integration easier