Trust Domain Extensions

- **Trust Domains**
  - Hardware-isolated virtual machines
  - Provide memory and CPU state confidentiality and integrity
  - Maintain CSP control of resources and platform integrity

- **Hardware + Software**
  - VMX / ISA extensions
  - Memory Encryption w/ Integrity
  - CPU-attested software module

- **In-Depth Presentation at Linux Security Summit**
  - https://sched.co/eCgM
Hardware

- Secure Arbitration Mode (SEAM)
  - New architectural CPU mode
  - Enter/Exit via SEAMCALL / SEAMRET
    - Transition between VMX Root and SEAM Root
  - VTx supported in SEAM
    - Transition between SEAM Root and SEAM Non-Root

- Multi-Key Total Memory Encryption w/ Integrity (MKTME-i)
  - Integrity failures cause #MC (recoverable)
  - Partition Key IDs into shared and private
    - Private Key IDs can only be used in SEAM
    - Shared bit in GPA (bit 51 or 47) selects shared vs. private

- Shared EPTP
  - Second EPTP added to VMCS for shared memory management
  - Shared bit in GPA (bit 51 or 47) selects shared vs. private/secure
Software

- Trust Domain Extensions (TDX) Module
  - Intel developed module that runs in SEAM
  - TDX-Module manages private guest state
    - Context switches register state, XSAVE state, MSRs, etc...
    - Directly controls S-EPT, VMCS, etc...
    - Reflects instruction-based VM-Exits as #VEs
  - Exposes ABI to VMM to create Trust Domains (TD)
  - VMM manages resources, e.g. memory usage, scheduling, etc...

- SEAM Loader
  - Authenticated Code Module (ACM) that loads TDX-Module
    - TDX-Module protected via SEAM range register (SEAMRR)
  - Configures SEAM VMCS for SEAMCALL
Touchpoints

- **Boot**
  - Launch SEAM Loader ACM (BSP)
  - Configure TDX-Module (all CPUs)

- **Core KVM**
  - Wrap x86 ops callbacks to achieve VMX/TDX coexistence
    - No meaningful performance impact to VMX or SVM
  - Reuse select portions of VMX
    - IRQ/NMI handlers, Posted Interrupt support, EPT entry points, etc...
  - Moderate refactoring to x86 and common KVM
    - Piggyback and repurpose SEV’s ioctl()s
    - TDX-Module API ordering doesn’t perfectly align with KVM

- **MMU**
  - Non-trivial KVM MMU changes to support S-EPT
  - Kernel MMU support to unmap guest private memory
MMU

- **Shared vs. Private Memory**
  - Alias shared->private GPAs in memslots
    - Treat Shared bit as an attribute bit
    - Disallow shared bit in “real” memslot GPA
  - Hide shared bit from host userspace
    - Ignore/strip shared bit for emulated MMIO TDVMCALLs

- **Secure EPT (S-EPT)**
  - MMU hooks to insert/zap/remove S-EPT entries
  - Maintain shadow copy of S-EPT tables
    - SEAMCALL is very expensive
    - Memory for page tables is relatively cheap
  - Additional API to create S-EPT translations without a page fault
...MMU

- **Private Memory**
  - Private memory must reside in a Trust Domain Memory Region (TDMR)
    - Kernel adds all RAM to TDMR array at boot
    - KVM allocates private memory as normal, “gifts” to TDX-Module / TD
    - HugeTLBFS, THP, memfd, anon, etc.... all supported
  - Unmap guest private memory (not yet implemented)
    - Prevent userspace from inducing integrity failures, i.e. #MCs

- **EPT Violation #VE**
  - Configure shared EPT to reflect emulated MMIO as #VE
  - #VE suppression is opt-out (non-zero init value for EPTEs)

- **Advanced Features (not yet implemented)**
  - 2mb/1gb S-EPT large pages
  - Host page migration, e.g. NUMA balancing
  - Page promotion/demotion
Status

- “Basic” Functionality
  - KVM code complete, QEMU functional
  - Kernel - 40+ files, 7,000+ insertions(+), ~700 deletions(-)
  - [https://github.com/intel/tdx](https://github.com/intel/tdx) kvm

- Near Future (prior to upstreaming)
  - Large page support
  - Host page migration
  - Unmap guest private memory

- Less Near Future
  - Live Migration
  - Nested Virtualization
Light Reading

Thank You!