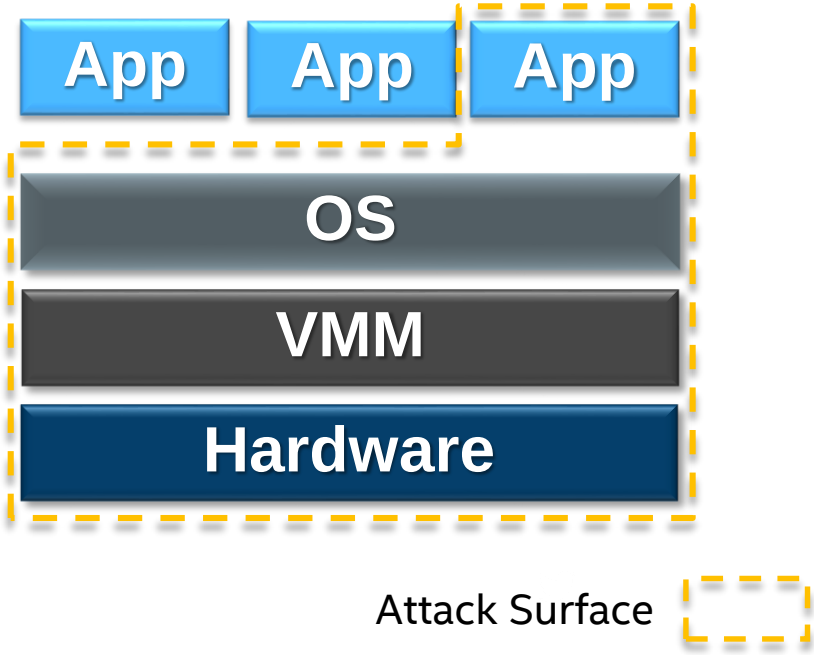
Sean Christopherson
Intel

Intel SGX Virtualization

KVM Forum 2018

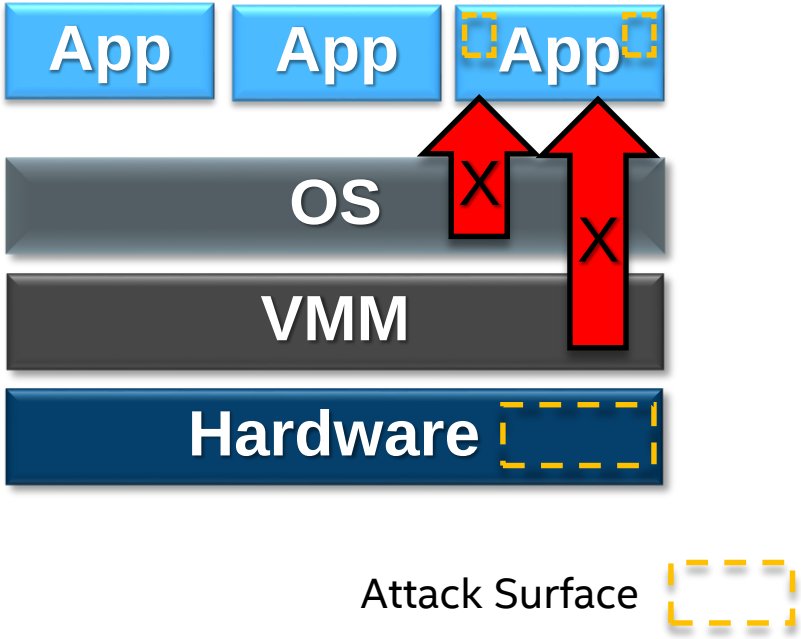
Traditional VM Landscape

- App's secrets accessible by any privileged entity, e.g. VMM and OS
- … or a malicious app that has exploited flaws to escalate privileges
- Encrypting VM's memory doesn't move OS/VMM/Firmware out of TCB



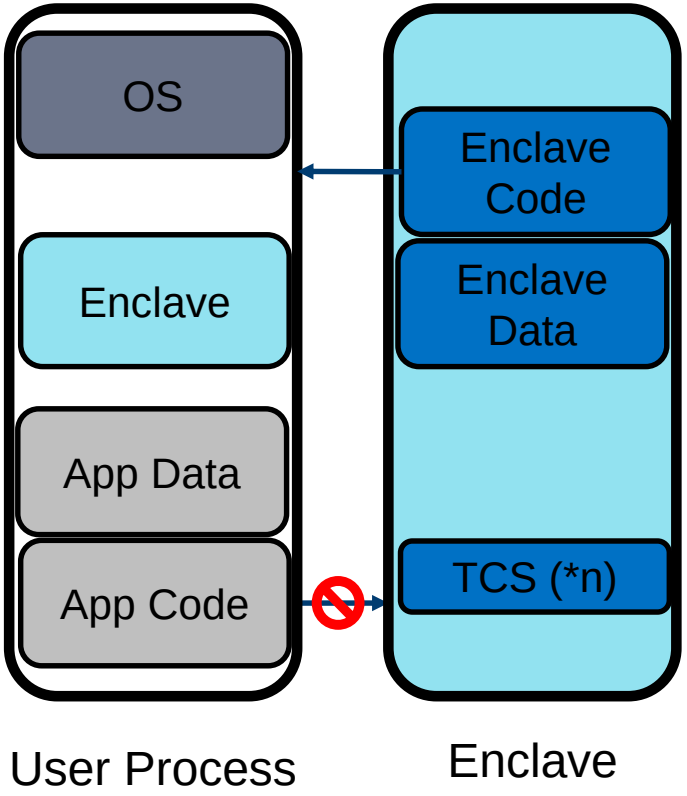
How do Software Guard eXtensions help?

- Reduced attack surface
- App's secrets are protected even if VMM, OS, BIOS, etc… are subverted
- Enclave can attest itself to 3rd party with H/W root of trust
  - What's running in the enclave
  - What's the execution environment
  - What are the CPU's security properties
  - And other stuff…



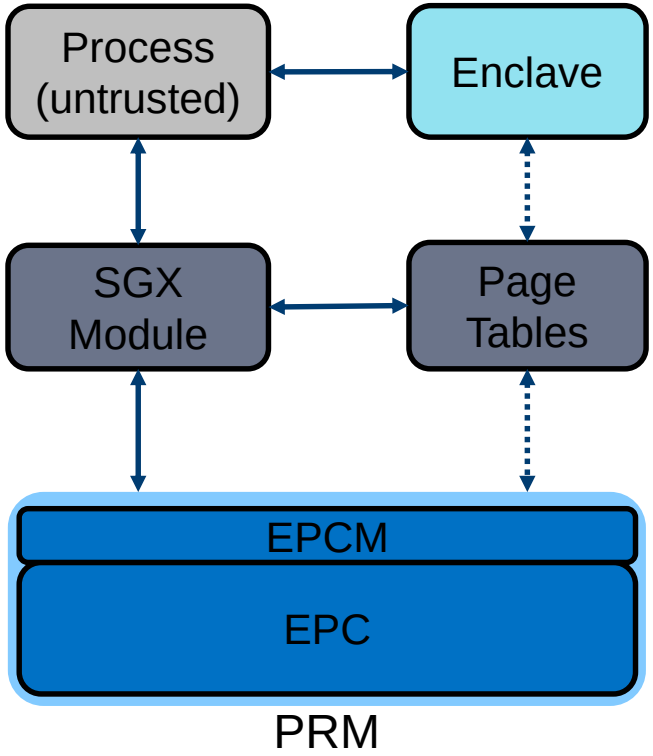
SGX Enclaves

- Trusted execution environment embedded in a process
- Separate code and data, with controlled entry points
- Multi-threading via Thread Control Structures (TCS)
- Enclave has full read/write access to process' virtual memory (no exec)
- … but not the other way 'round

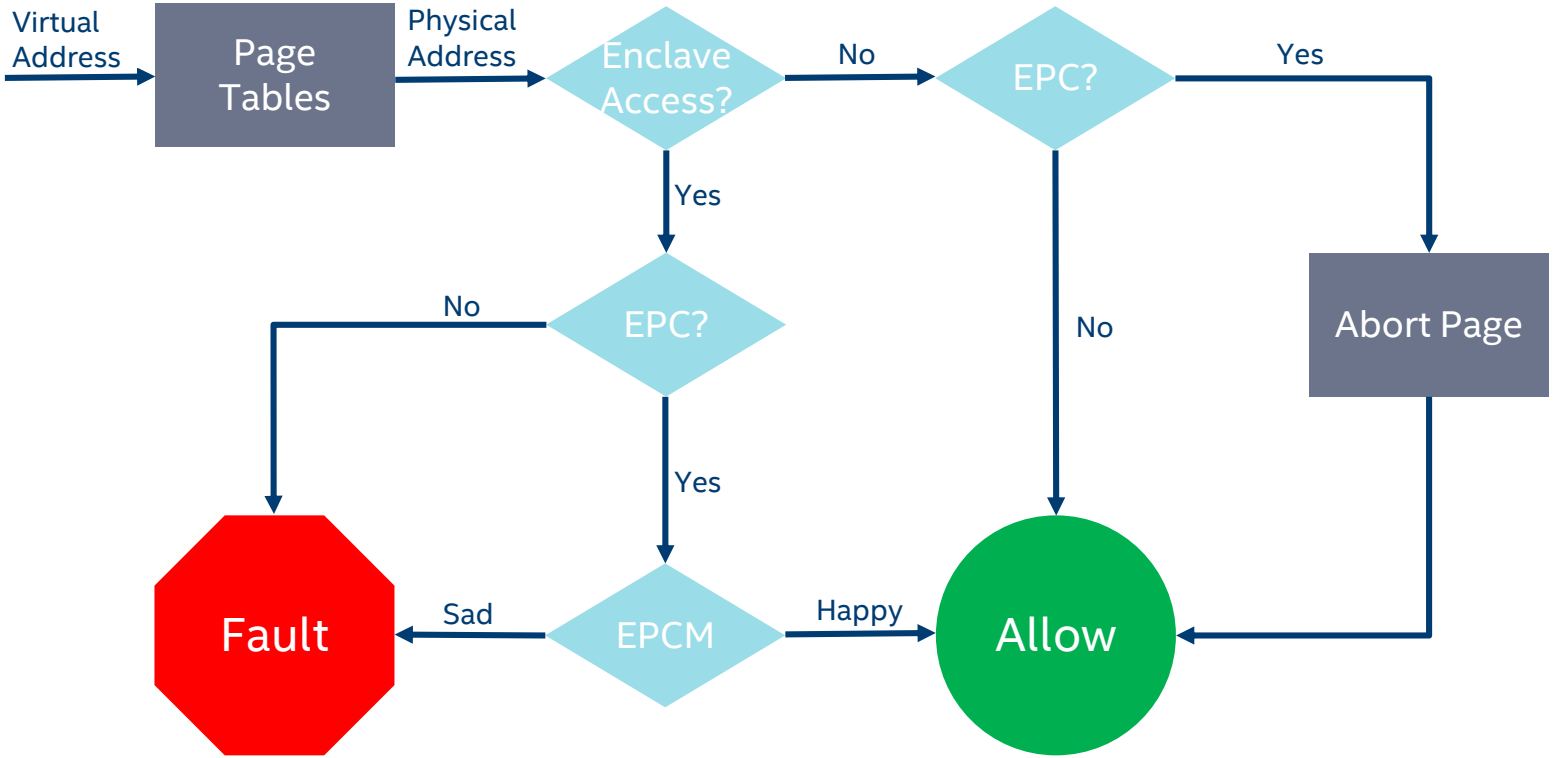


SGX High-Level View

- Userspace Instructions (ENCLU): EENTER, ERESUME, EEXIT, etc...
- Kernel Instructions (ENCLS): ECREATE, EADD, EINIT, EREMOVE, EWB, etc...
- Hardware: Processor Reserved Memory (PRM), Enclave Page Cache (EPC) and EPC Map (EPCM)

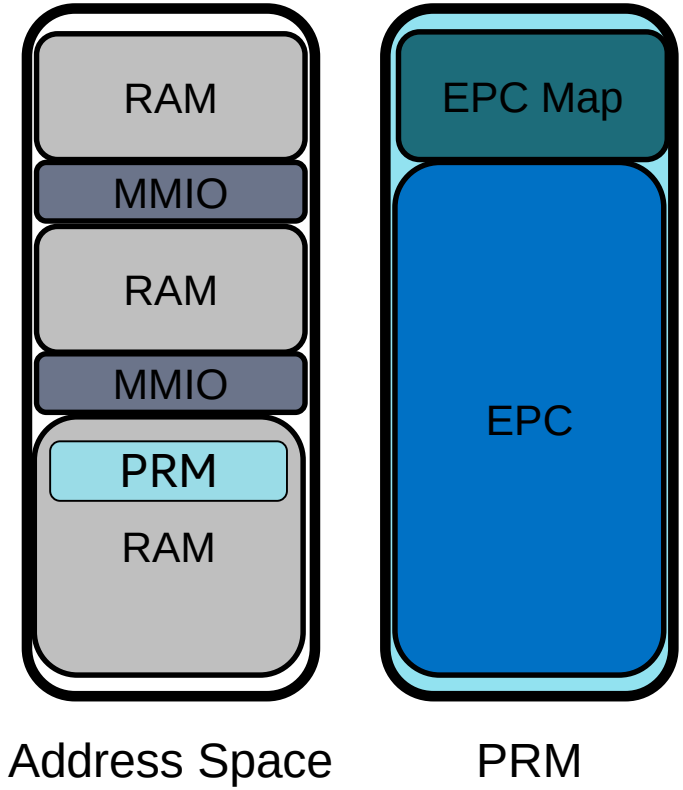


SGX Access Control



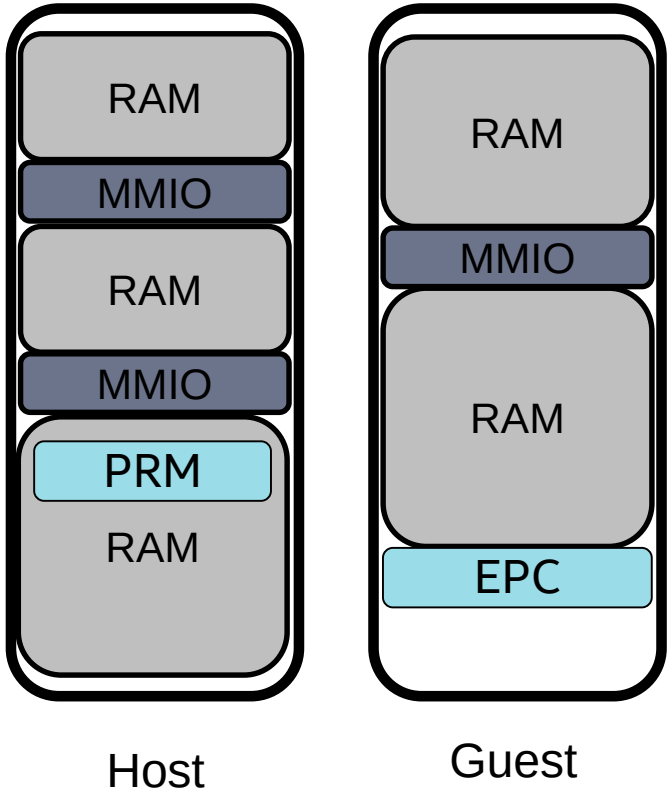
EPC - Bare Metal

- PRM carved out of RAM via range registers
  - Statically partitioned and locked at boot
  - Power-of-2 sized, naturally aligned
  - EPCM uses percentage of PRM
- PRM is encrypted with ephemeral key
  - Transparently {de,en}crypted on read/write from/to DRAM (unencrypted in CPU cache)
  - New key generated by CPU at reset
  - EPC{M} is zapped if CPU powers down
    - All EPCM entries marked invalid
    - Kernel/Userspace must handle faults



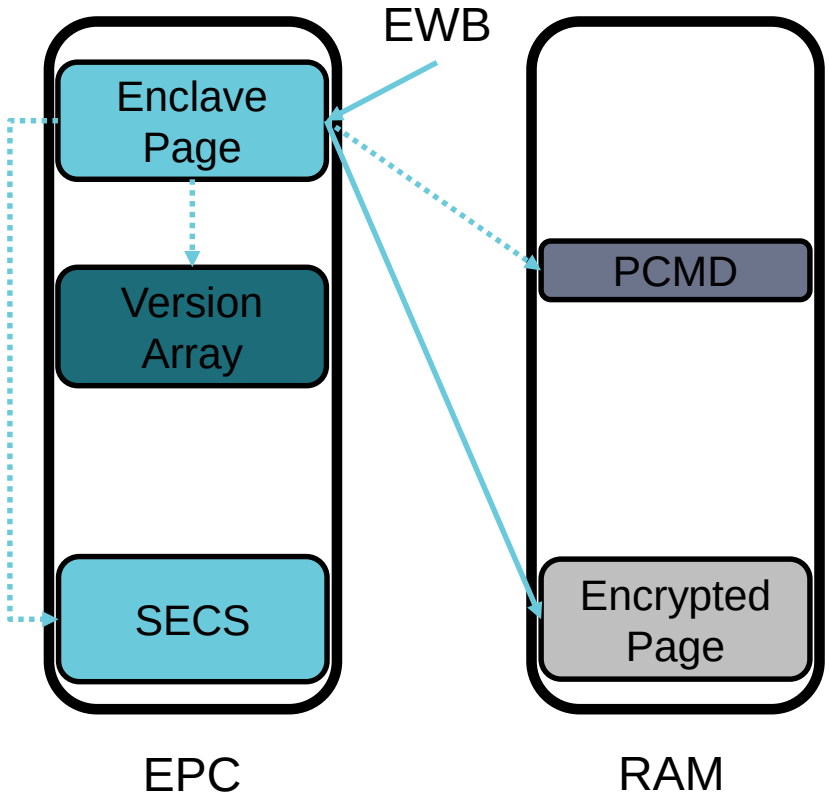
EPC - Virtual Machine

- No PRM, no EPCM
- Virtual EPC is less restricted
  - Doesn't need to be backed by guest RAM
  - Can be 4k page sized/aligned
- VMM can exploit loss of EPC{M} behavior
  - Migration!
  - Pseudo-reclaim
  - Other tricks?



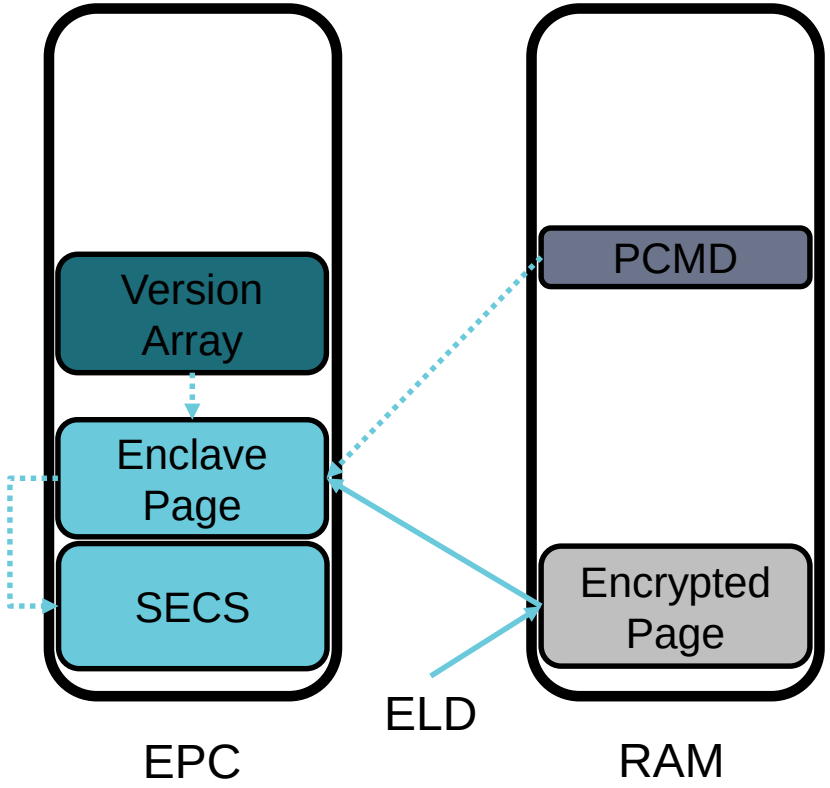
EPC - Page Out

- EWB: Enclave Write-Back
  - Write encrypted data to e.g. RAM
  - Write ID to version array
  - Write metadata to PCMD
- EWB decrements SECS refcount
  - SECS with children can't be evicted
- All page types can be evicted



EPC - Page In

- ELD: Enclave Load
  - Load encrypted data from e.g. RAM
  - Verify metadata from PCMD
  - Verify ID from version array
- ELD increments SECS refcount



SGX Launch Control

- EINIT token required to initialize an enclave
  - EINIT token can only be generated by Launch Enclave
  - Launch Enclave doesn't require token, but must be signed by LE key
  - Initial hardware (e.g. Skylake) hardcoded the LE key to an Intel key
- Launch Control (LC)
  - Allows creation of enclaves without Intel's blessing
  - Provides four MSRs for user/kernel to specify LE public key
    - MSRs writable if FEATURE_CONTROL.SGX_LC enabled
    - MSRs also writable prior to SGX activation, i.e. by firmware
  - Presence of SGX LC enumerated via CPUID

SGX Virtualization Touchpoints

- CPUID
  - Feature bits in leaf 0x7 for SGX and SGX LC
  - New SGX-specific leaf, 0x12, with 2+ sub-leafs
    - Sub-leaf 0 enumerates instruction sets (SGX1, SGX2, etc...)
    - Sub-leaf 1 enumerates supported SECS.ATTRIBUTES bits
    - Sub-leafs 2+ enumerate EPC sections, a.k.a. EPC memory regions
- MSRs
  - SGX and SGX_LC bits in feature control
  - LE public key hash MSRs

SGX Virtualization Touchpoints Cont…

- ENCLS-exiting VMCS field
  - Per-leaf controls to intercept ENCLS leafs
  - New VMExit reason, EXIT_REASON_ENCLS
- Enclave Page Cache
  - Expose virtual EPC region to guest
  - Manage physical backing of EPC
  - Reclaim and oversubscription (here be dragons)

KVM - ENCLS

- Intercept ENCLS leafs to inject #UD and #GP as necessary
  - No CR4 enable bit for SGX (#UD)
  - Leafs can be disable via CPUID bits (#UD)
  - SGX can be disabled in Feature Control MSR (#GP)
- Pass-through guest-supported SGX1 and SGX2 ENCLS leafs
- … unless Launch Control is enabled (in host)
  - WRMSR(SGXLEPUBKEYHASHn) is **slow** (~400 cycles per MSR, 4x MSRs)
  - EINIT is even slower (70k+ cycles) and interruptible
  - Intercept and execute EINIT w/ guest's LE public key hash
- Merge with L1's ENCLS-exiting bitmap to support nested SGX

KVM - Enclave Page Cache

- Implementation
  - Same basic approach as RAM, e.g. allocate on fault/access
    - Adjust VMA to attach fault handler and tweak flags, e.g. VM_PFNMAP
    - Allocate EPC pages from SGX subsystem, insert PFN into host PTEs
  - Optionally reserve at VM creation, e.g. for migration (module param?)
  - No dependency on host userspace SGX driver (only SGX subsystem)
- Userspace API
  - Option 1 - Extend KVM_SET_USER_MEMORY_REGION w/ new EPC flag
    - Minimal changes to KVM
  - Option 2 - New ioctl() to specify EPC region(s)
    - Easier to extend in the future, e.g. per-VM reservation/oversubscription policies

Qemu - SGX Virtualization

- CPUID
  - Feature bits controllable by user, e.g. SGX, SGX_LC, SGX1, SGX2, etc...
  - Allowed SECS.ATTRIBUTES pulled from hardware
    - Can expose to user only if KVM intercepts ECREATE
    - Might be required for migration?
  - Expose virtual EPC section(s) to guest
- MSRs
  - SGX and LC bits set in fw_cfg.feature_control when possible
  - Defer to guest firmware for locking down LE hash MSRs (and feature control)

Qemu - EPC Virtualization

- New machine options ‘epc’ and ‘epc_below_4g’
  - epc=<size>: define size of virtual EPC in 4k chunks (page granularity)
  - epc_below_4g=<on|off|auto>: control placement of EPC
    - auto: allocate below 4g if possible, fallback to above 4g
    - off: allocate above 4g
    - allocate below 4g, report error if not possible
  - Location of virtual EPC exposed via CPUID and ACPI
- Migration allowed, but EPC is “lost”
  - EPC is tied to physical CPU, even if evicted
  - EPCM naturally generates faults after migration (EPCM entries invalid)
- Currently no mechanism to release EPC back to host

What About EPC Oversubscription?

- Hardware enforces strict EPC concurrent access rules
  - Avoiding conflicts is *painful* without additional ISA
  - Conflicts are visible to guest and cause faults in host
- EPCM refcounts SECS based on child pages
  - VMM can't evict SECS if its children are resident in EPC
  - VMM can't reload evicted pages if guest evicts SECS
- VMM EPC Oversubscription is complex (even by SGX standards)
- TL;DR: not supported in KVM, yet...

But The Word Yet…

- VMX Features for EPC Oversubscription on future hardware (in SDM now)
- New leafs to avoid faults on conflicts (ENCLS_C extensions)
- New ENCLV instruction to virtualize select SECS behavior
  - E{DEC,INC}VIRTCHILD: Prevent guest from evicting SECS
  - ESETCONTEXT: Fudge the back-pointer of a reloaded SECS
  - ENCLV-Exiting and EXIT_REASON_ENCLV to allow nesting
- New VMExit reason to handle EPC conflicts, EXIT_REASON_SGX_CONFLICT
  - Triggered when EPC conflict occurs in guest
  - Allows squashing faults/errors that may have been induced by VMM

EPC cgroup

- Motivation
  - EPC is a limited, shared system resource
    - SGX subsystem does not limit or prioritize EPC consumption
    - Swapping pages in and out of the EPC is expensive
    - Misbehaving or poorly written enclave can essentially cause SGX DoS
  - Integration into existing memory cgroup is infeasible
- Design
  - Modeled after memory cgroup v2
  - Per-process accounting, not per-thread/task
  - Account everything, e.g. VA and SECS pages
  - Kill enclaves (or VMs) if necessary to honor hard limit

When Will Then Be Now?

- SGX subsystem
  - Upstreaming has been a bumpy road
  - Outside chance at making 4.21
- KVM and Qemu
  - Waiting on SGX subsystem
  - RFCs soon...
- EPC cgroup
  - Likely defer until KVM bits land upstream
  - RFCs?

Kick The Tires

- KVM, EPC cgroup and userspace driver
  - https://github.com/intel/kvm-sgx
- Qemu
  - https://github.com/intel/qemu-sgx

INTEL OPEN SOURCE TECHNOLOGY CENTER | 01.org

Acronyms

- SGX: Software Guard eXtensions
- PRM: Processor Reserved Memory
- EPC{M}: Enclave Page Cache {Map}
- SECS: Secure Enclave Control Structure
- TCS: Thread Control Structure
- LC: Launch Control
- LE: Launch Enclave
- PCMD: Paging Crypto MetaData

ENCLS Leafs - Enclave Management

- ECREATE: configure initial enclave environment
- EADD: add page to an uninitialized enclave
- EAUG: add page to an initialized enclave
- EEXTEND: extended the measurement of the enclave
- EINIT: verify and initialize enclave
- EDBG{RD,WR}: read/write from/to a debug enclave's memory
- EMODPR: restrict an EPC page's permissions
- EMODT: modify an EPC page's type

ENCLS Leafs – EPC Management

- EBLOCK: mark a page as blocked in EPCM
- ETRACK{C}: activate blocking tracing
- EWB: write back page from EPC to RAM
- ELD{B,U}{C}: load page in {un}blocked state from RAM to EPC
- EPA: add version array (to store evicted pages' metadata)
- EREMOVE: remove a page from EPC
- ERDINFO: retrieve info about an EPC page from EPCM (for virtualization)

ENCLV Leafs

- E{DEC,INC}VIRTCHILD: {dec,inc}rement SECS virtual refcount
- ESETCONTEXT: set SECS' context pointer

ENCLU Leafs

- Control Flow
  - EENTER: enter enclave at enclave-defined point
  - ERESUME: resume enclave at interrupted point
  - EEXIT: exit enclave (*without* wiping register state)
- Enclave Management
  - EACCEPT{COPY} : accept a new/changed EPC page
  - EMODPE: extend an EPC page's permissions
- Attestation, Provisioning and Sealing
  - EGETKEY: get cryptographic key
  - EREPORT: generate cryptographic report of the enclave

EPC cgroup User Interface

- epc.low – read-write, best effort epc protection
- epc.high – read-write, throttle limit on epc usage
- epc.max – read-write, hard limit on epc usage
- epc.current – read-only, displays current total epc usage
- epc.stat – read-only, displays detailed and historical statistics, e.g. # faults
- epc.events – read-only, notifies on significant behavior, e.g. reclaim when low