Emulating Hyper-V's Virtual Secure Mode (VSM) with QEMU and KVM

Nicolas Saenz Julienne

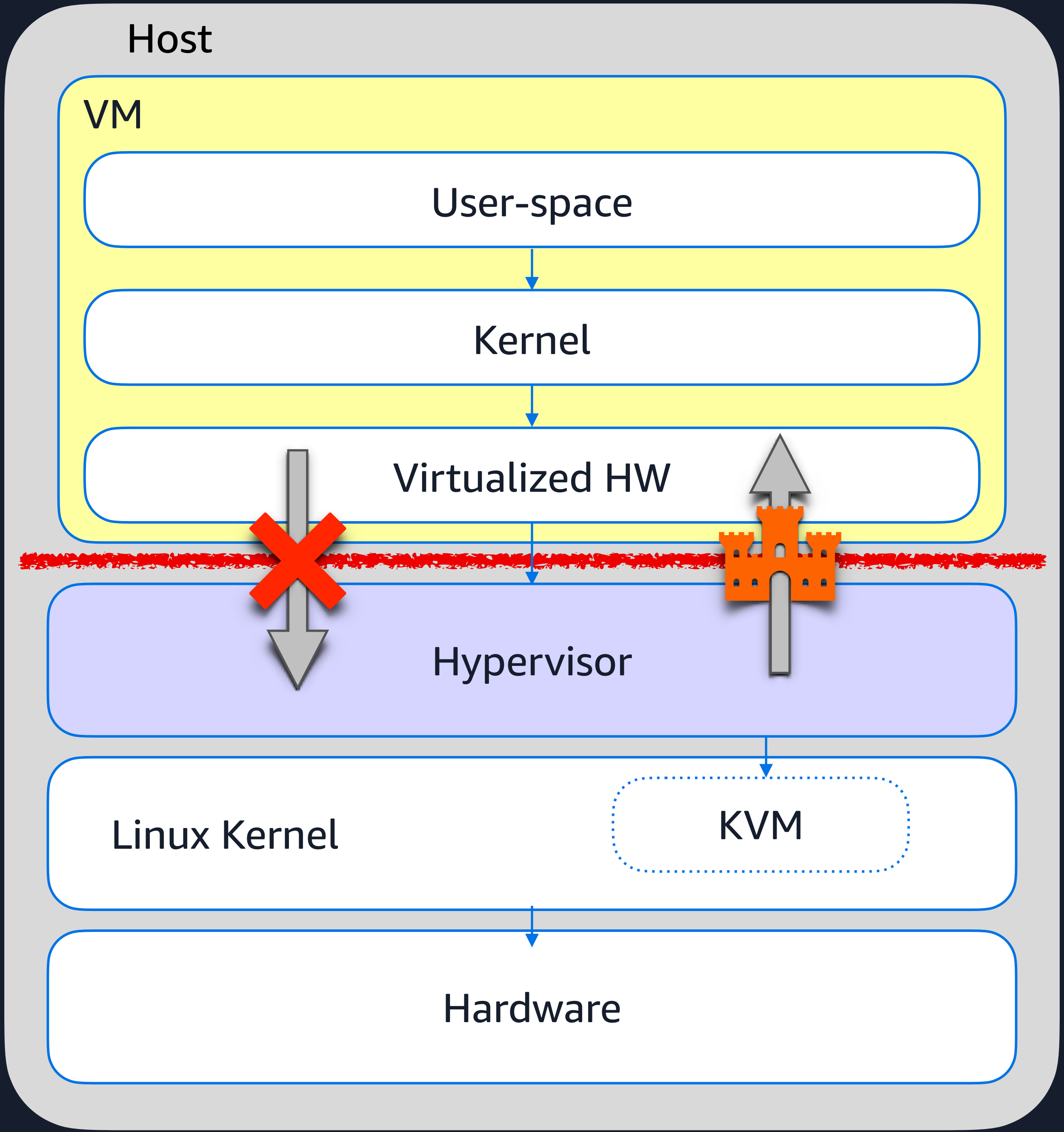
About me

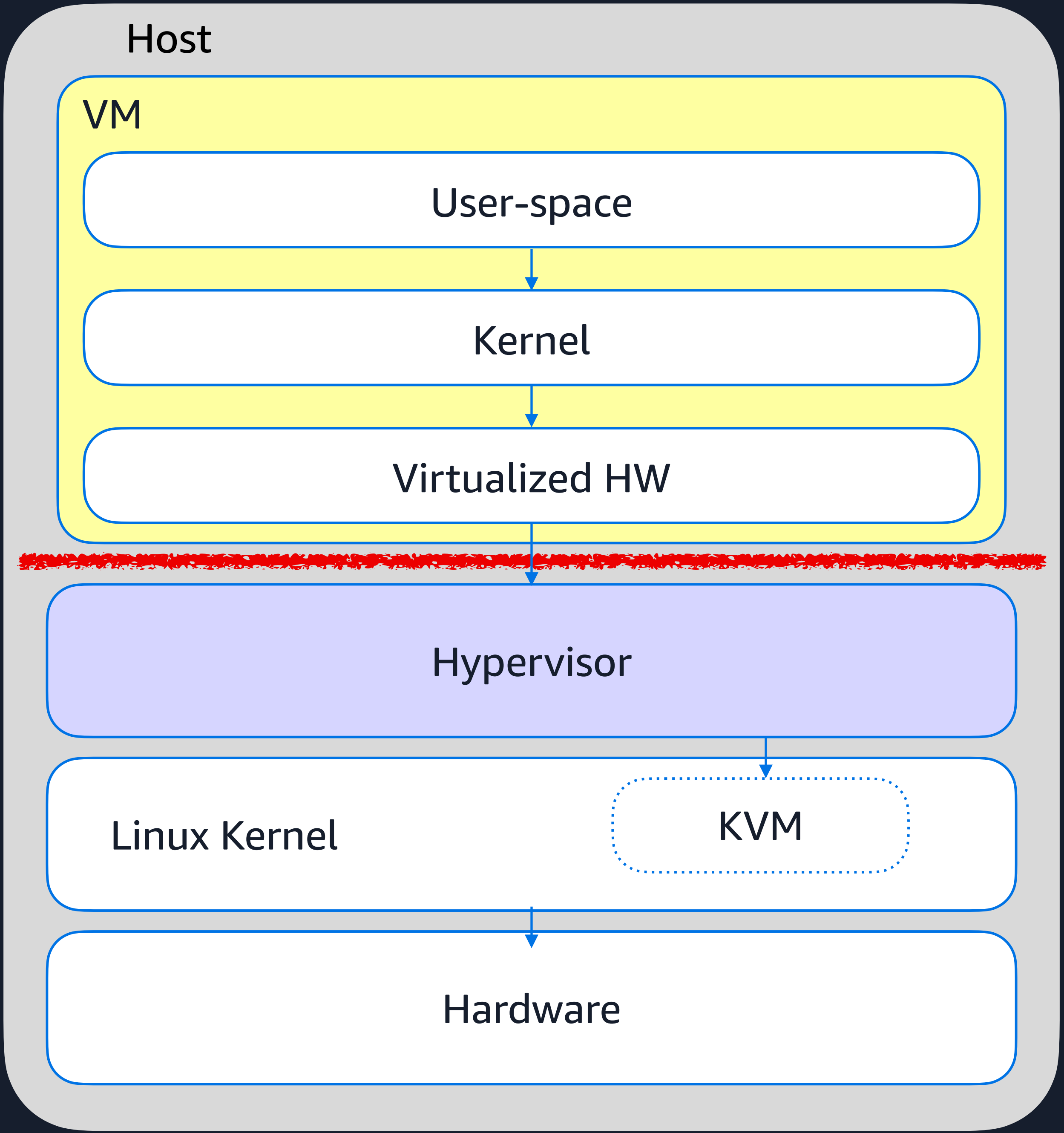
Nicolas Saenz Julienne

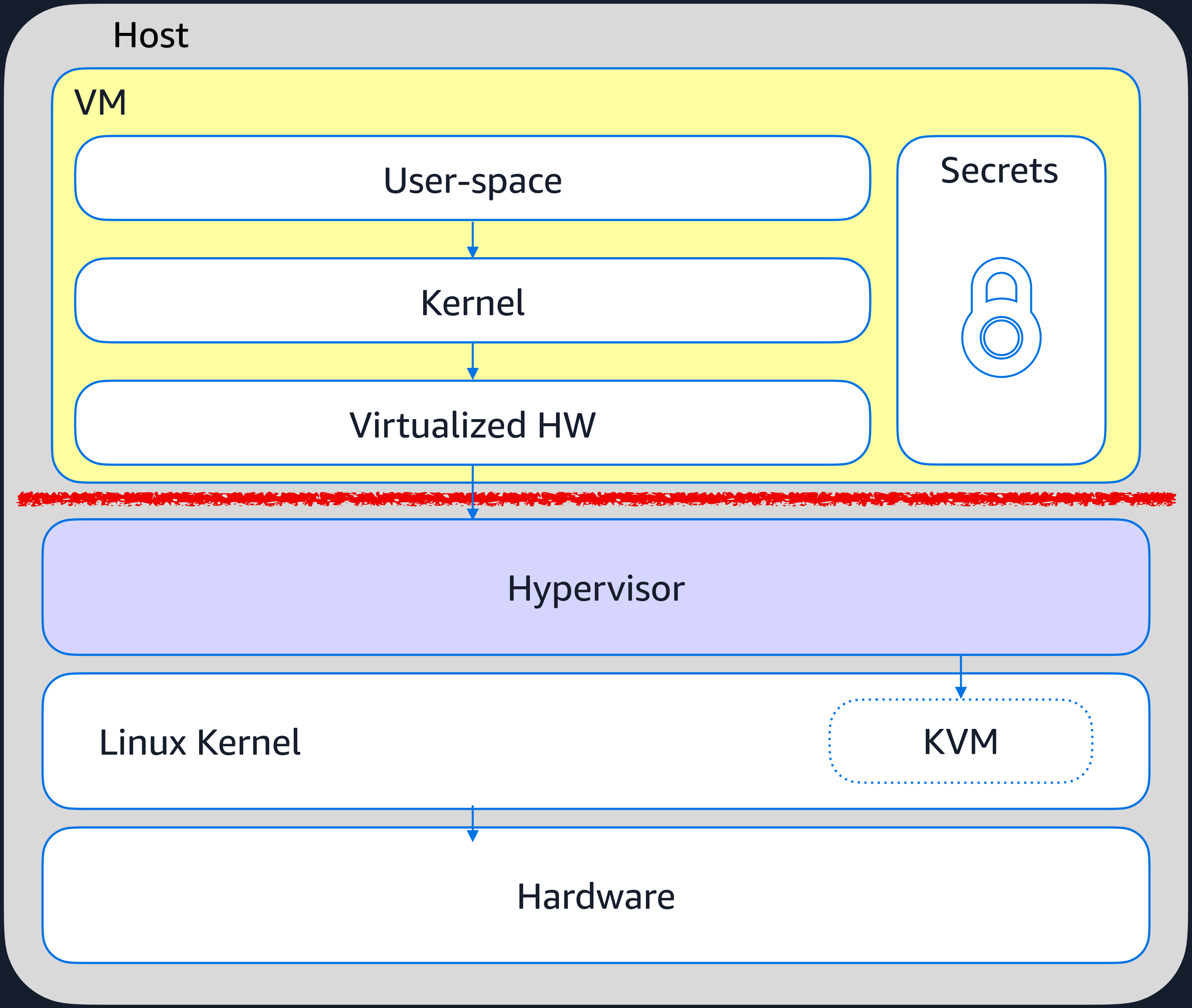
EC2 developer at Amazon

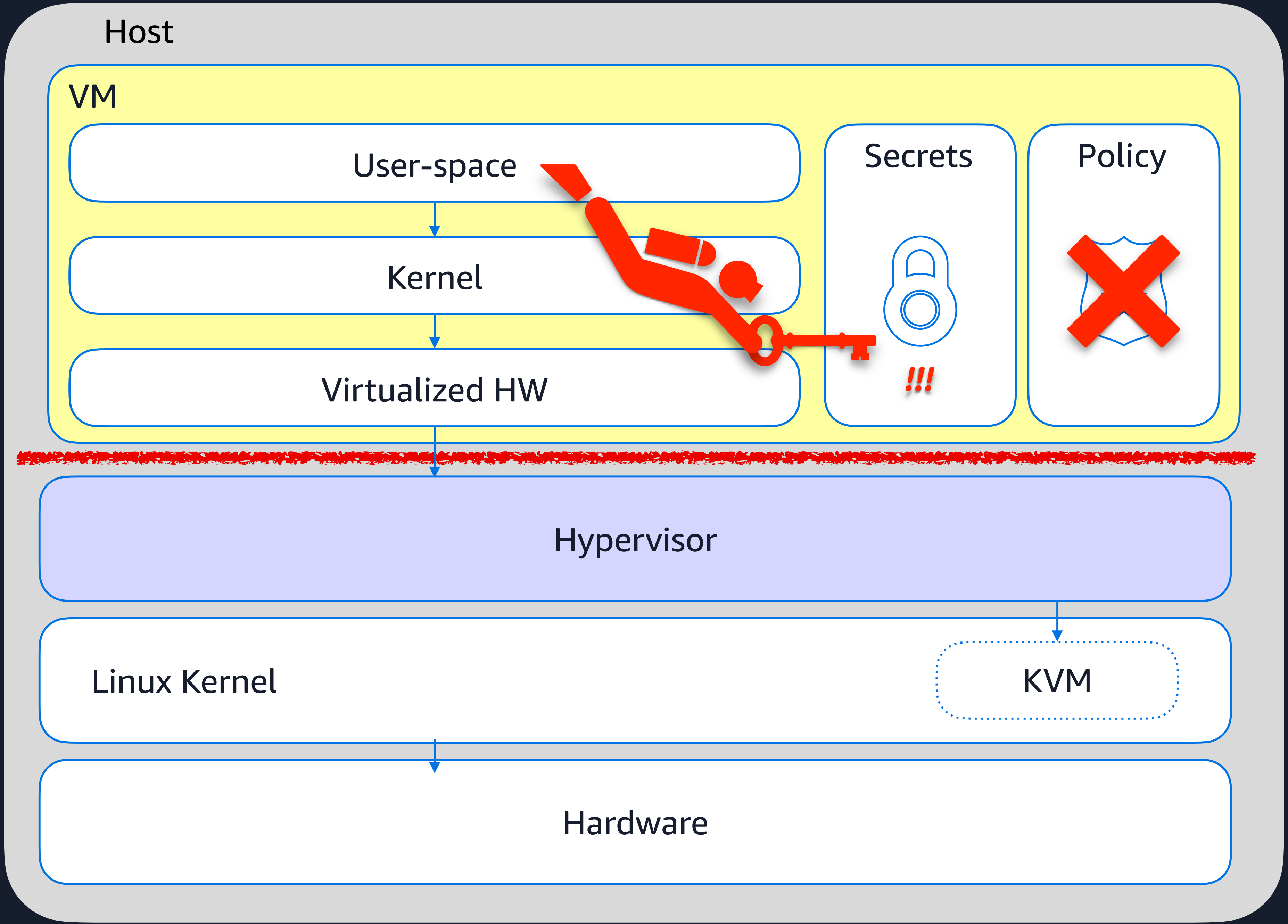
Opinions my own

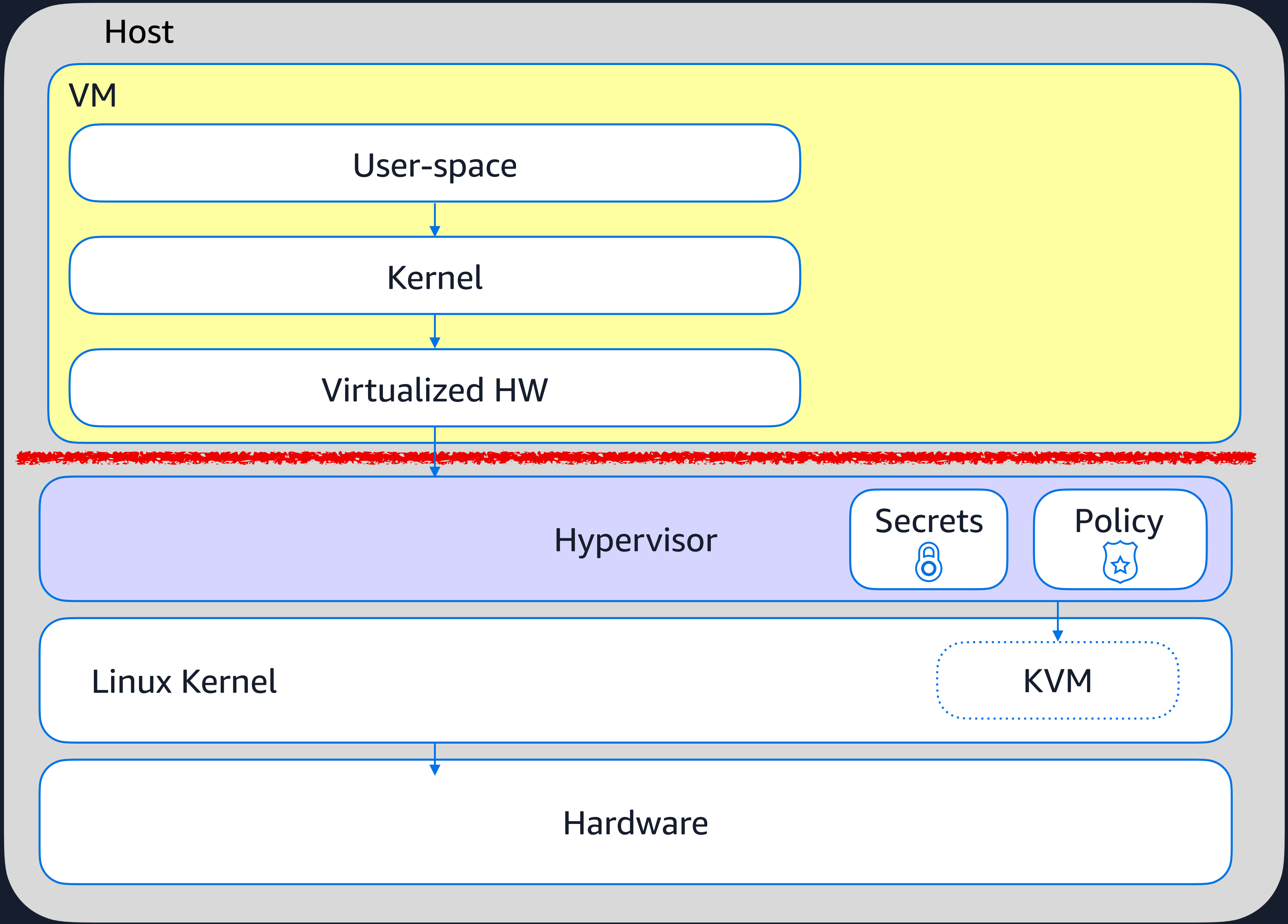
Virtualization Based Security (VBS)

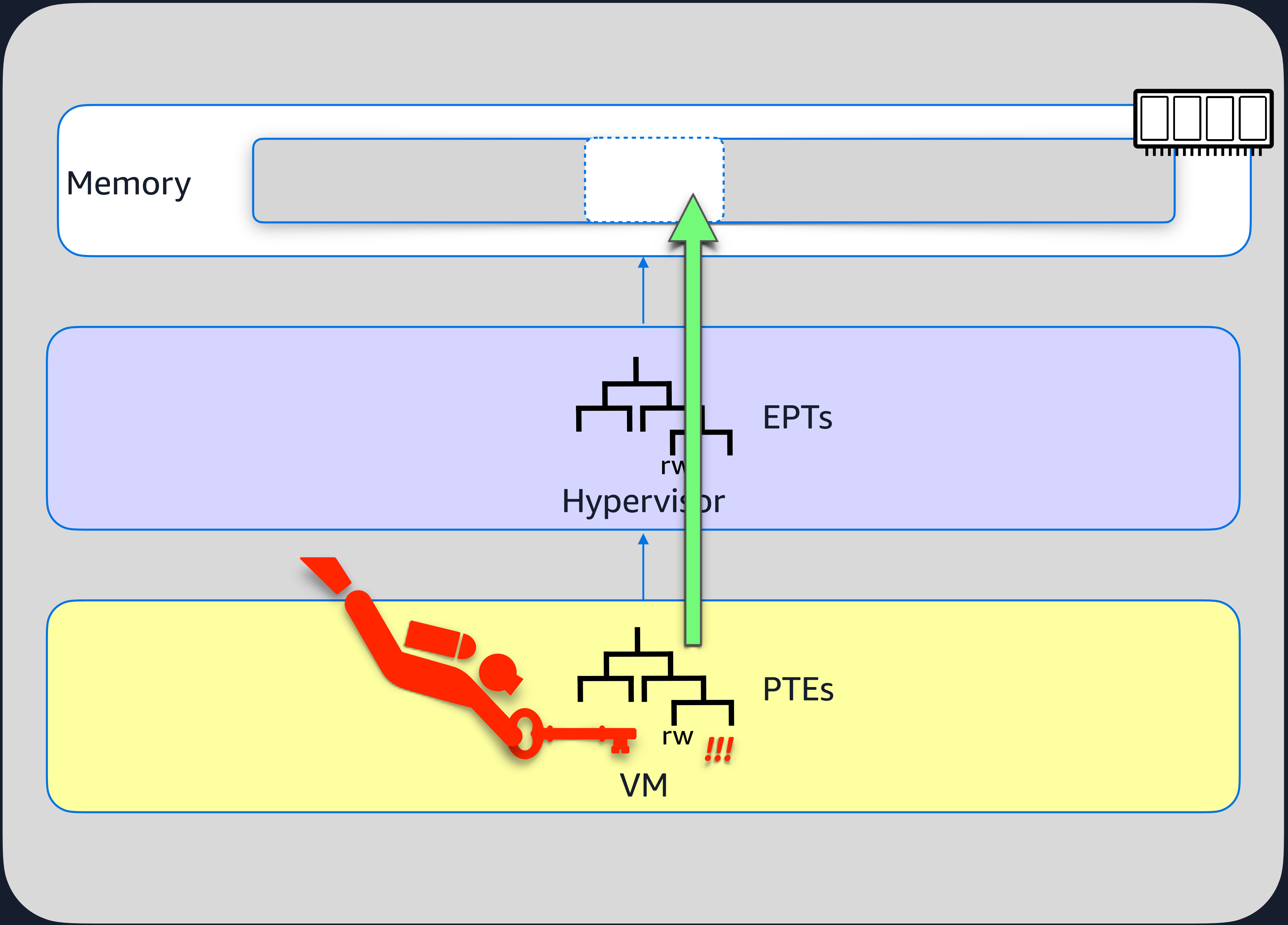

Host
VM
User-space
Kernel
Virtualized HW
Hypervisor
Linux Kernel
KVM
Hardware

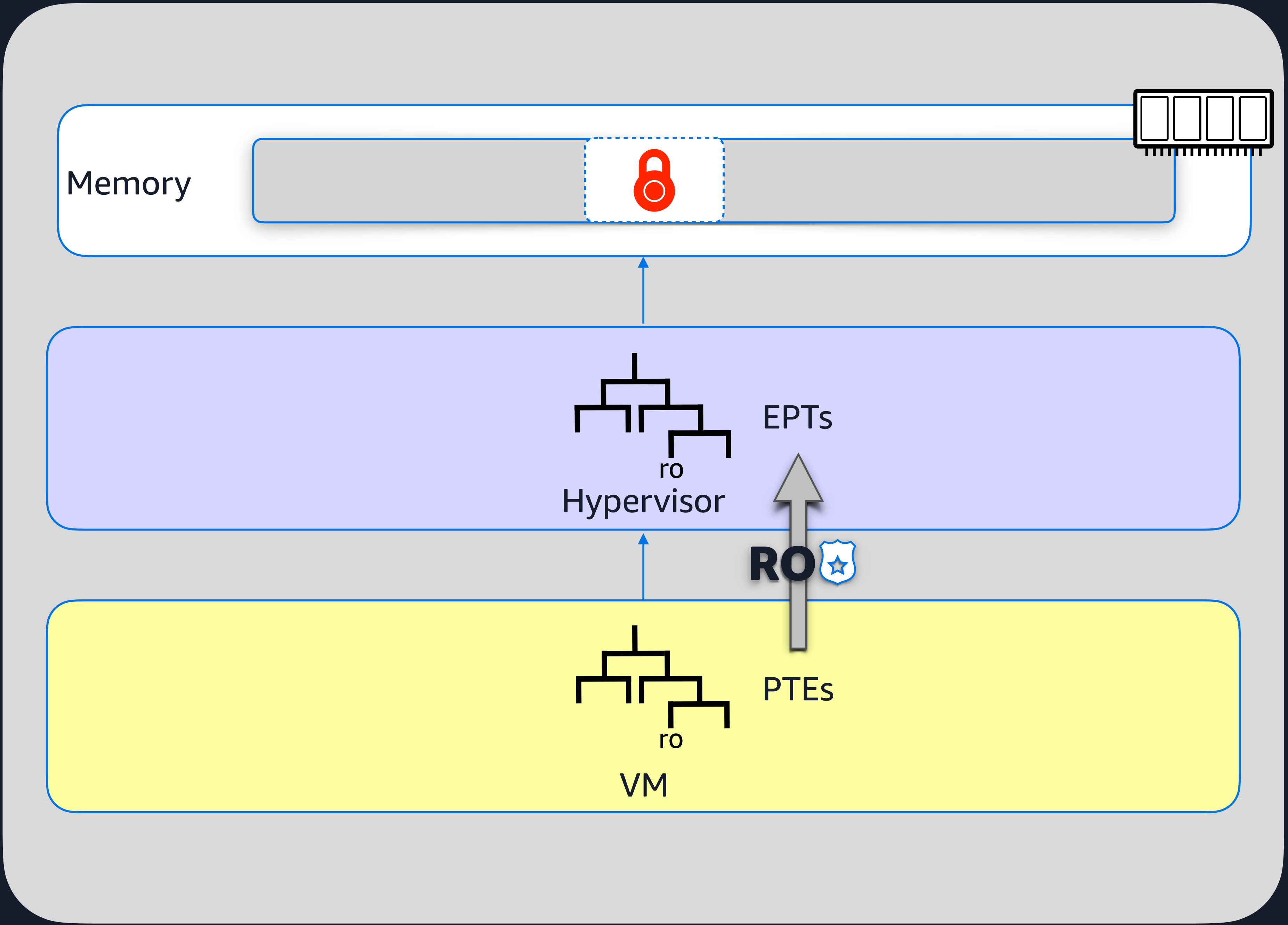

Host
VM
User-space
Kernel
Virtualized HW
Hypervisor
Linux Kernel
KVM
Hardware

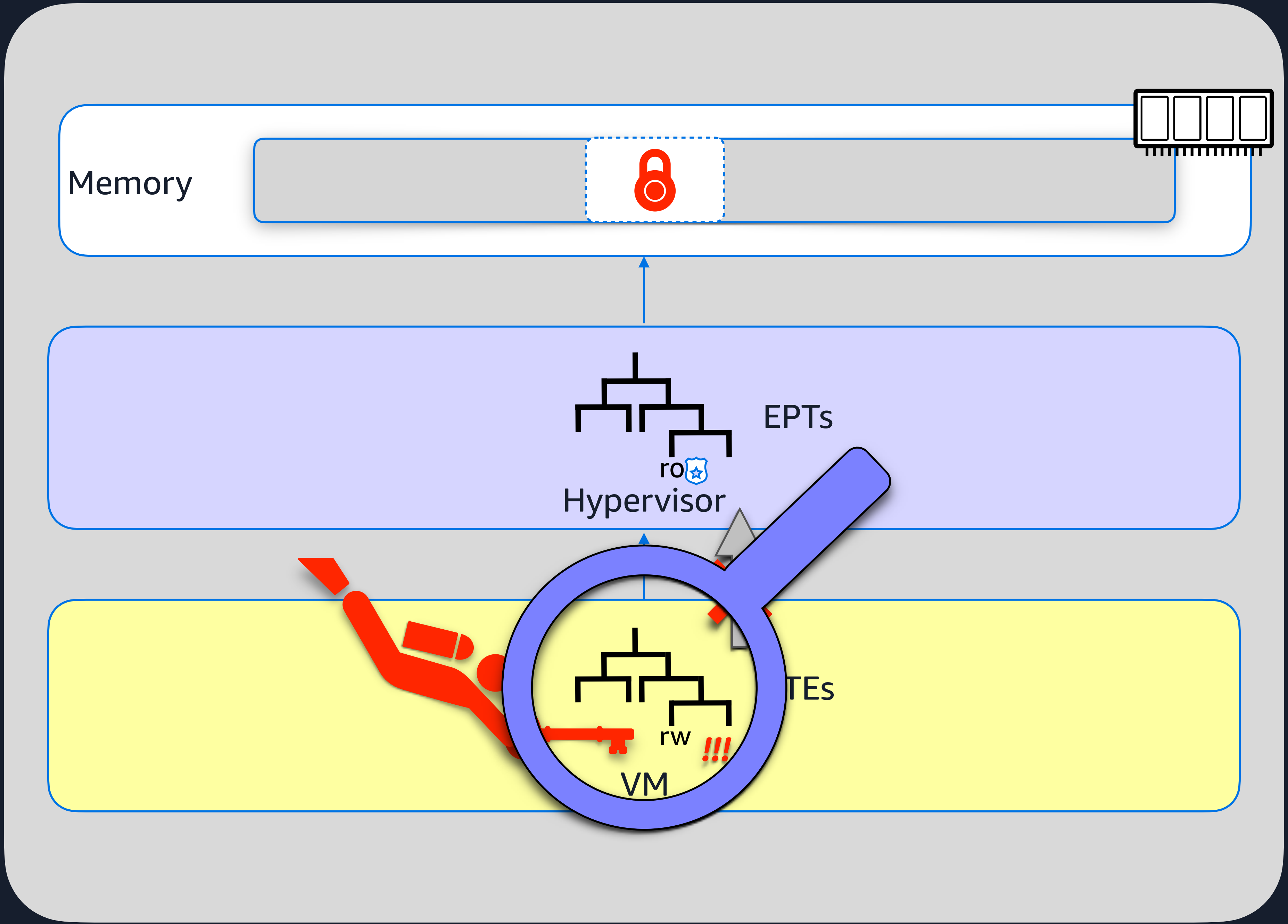

Host
VM
User-space
Kernel
Virtualized HW
Secrets
Hypervisor
Linux Kernel
KVM
Hardware

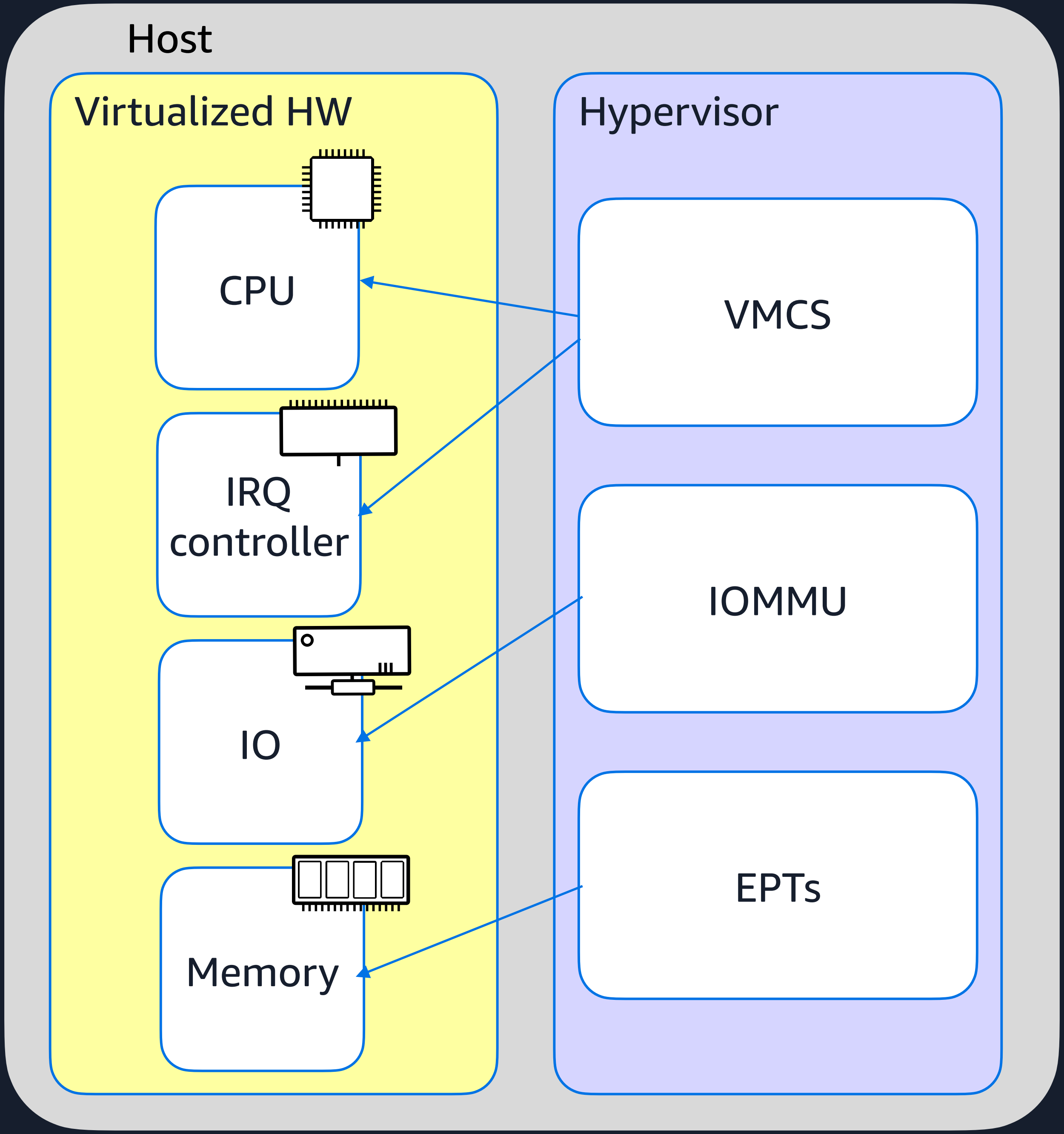

Host
VM
User-space
Kernel
Virtualized HW
Secrets
!!!
Policy
Hypervisor
Linux Kernel
KVM
Hardware


Host
VM
User-space
Kernel
Virtualized HW
Hypervisor
Secrets
Policy
Linux Kernel
KVM
Hardware

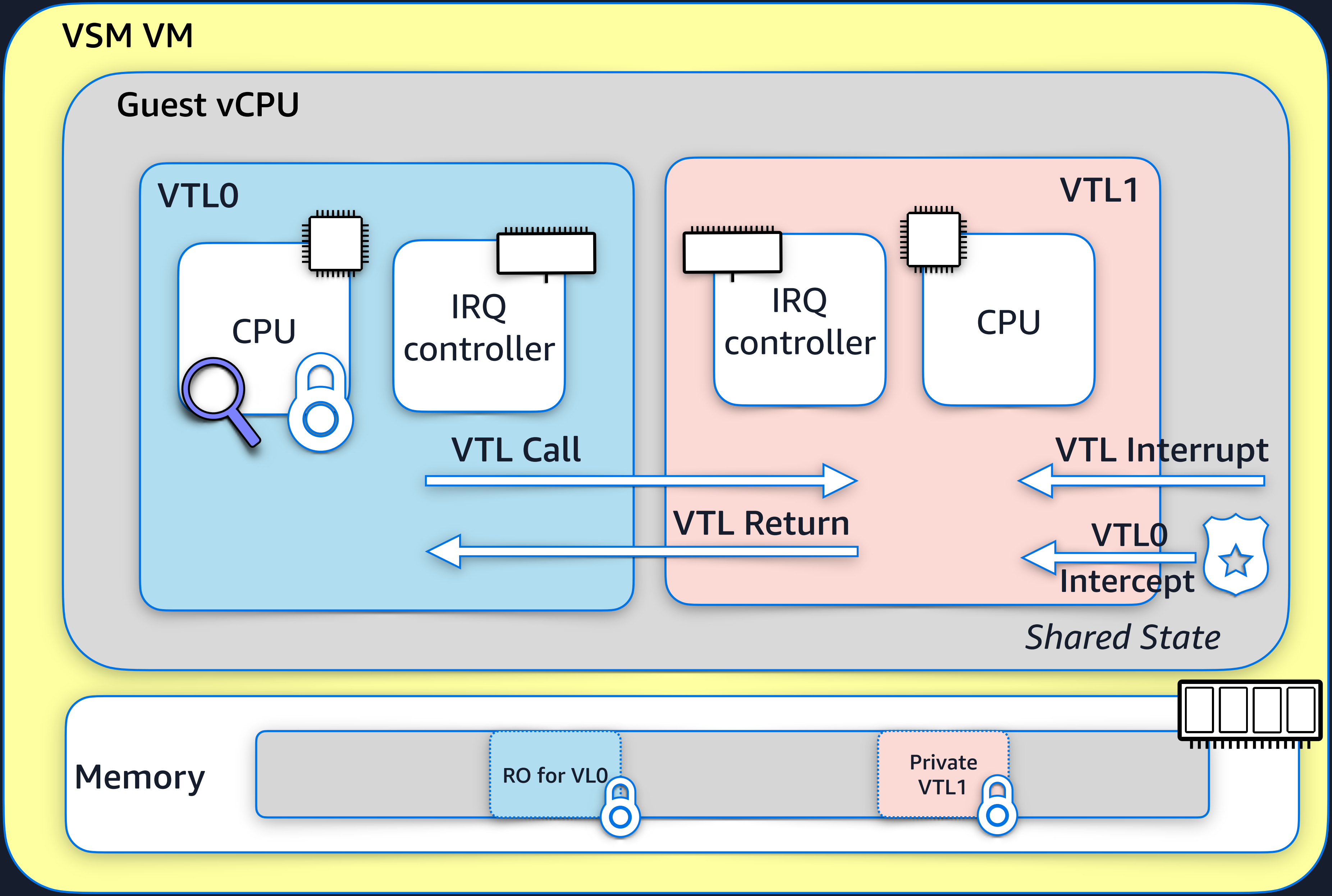

Memory
EPTs
rw
Hypervisor
PTEs
rw
!!!
VM


Memory
EPTs
ro
Hypervisor
RO
PTEs
ro
VM

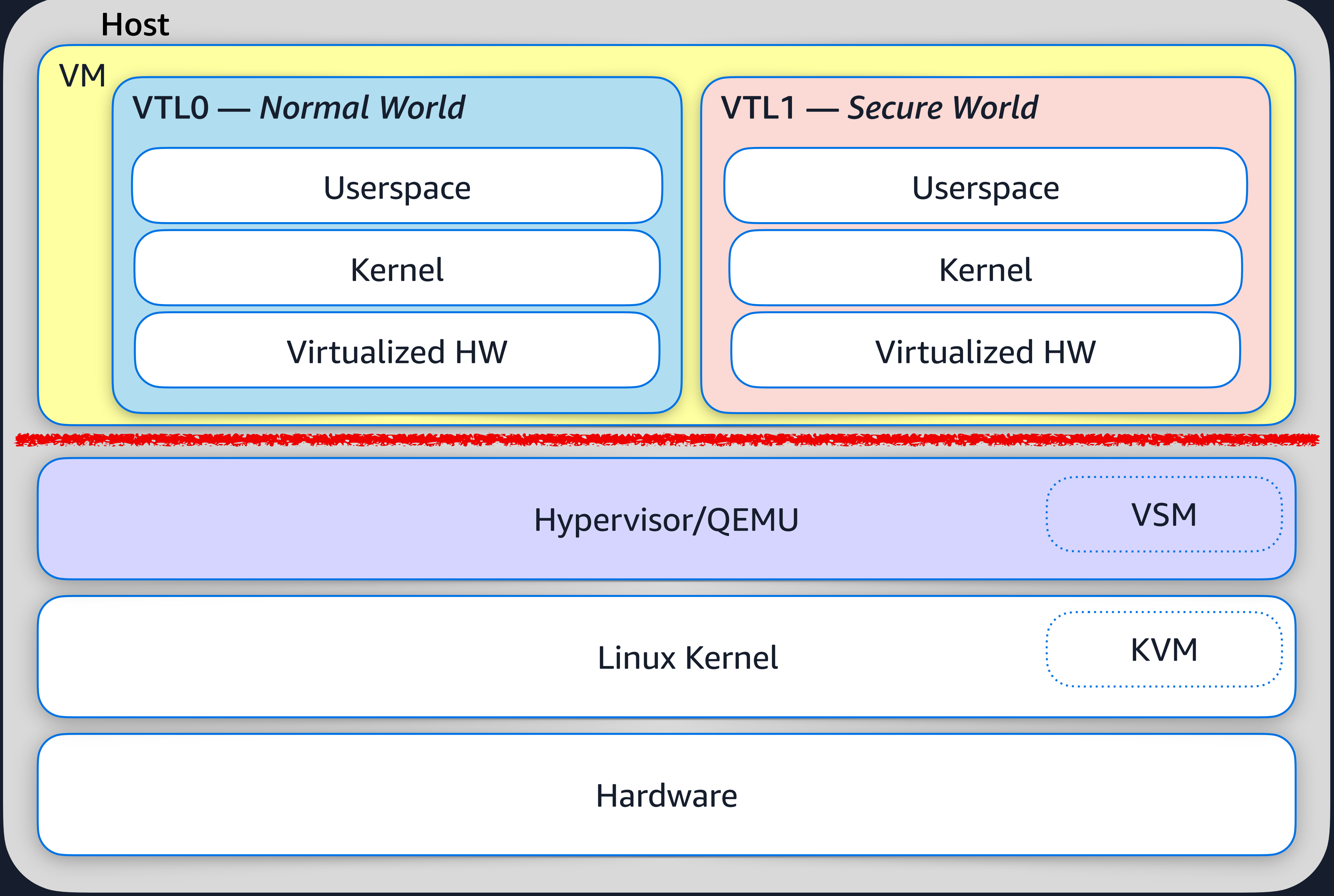

Memory
EPTs
ro
Hypervisor
TEs
rw
!!!
VM

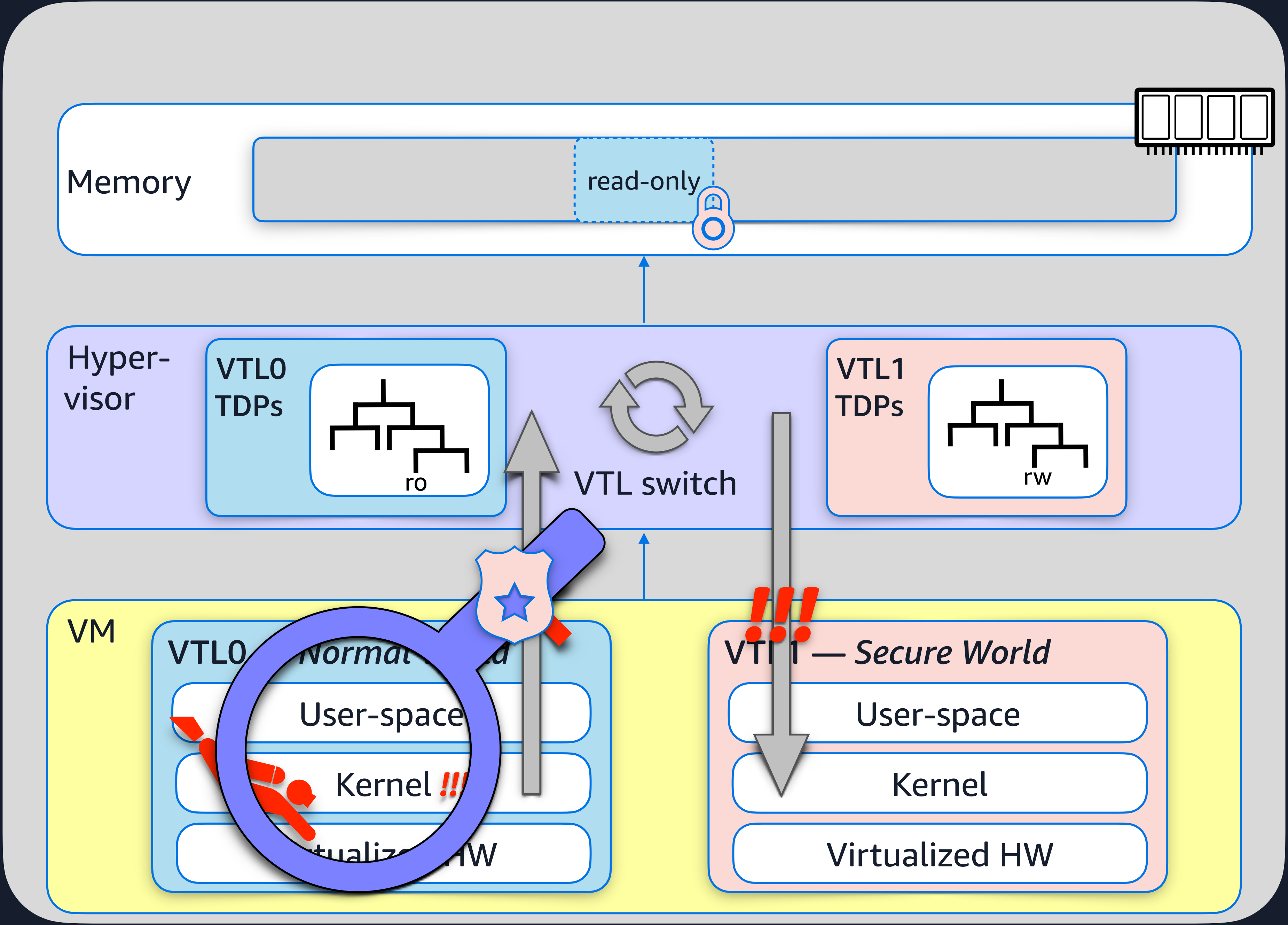

Host
Virtualized HW
CPU
IRQ controller
IO
Memory
Hypervisor
VMCS
IOMMU
EPTs

Hyper-V’s Virtual Secure Mode (VSM)


VSM VM
Guest vCPU
VTL0
CPU
IRQ controller
VTL1
IRQ controller
CPU
VTL Call
VTL Interrupt
VTL Return
VTL0 Intercept
Shared State
Memory
RO for VL0
Private VTL1

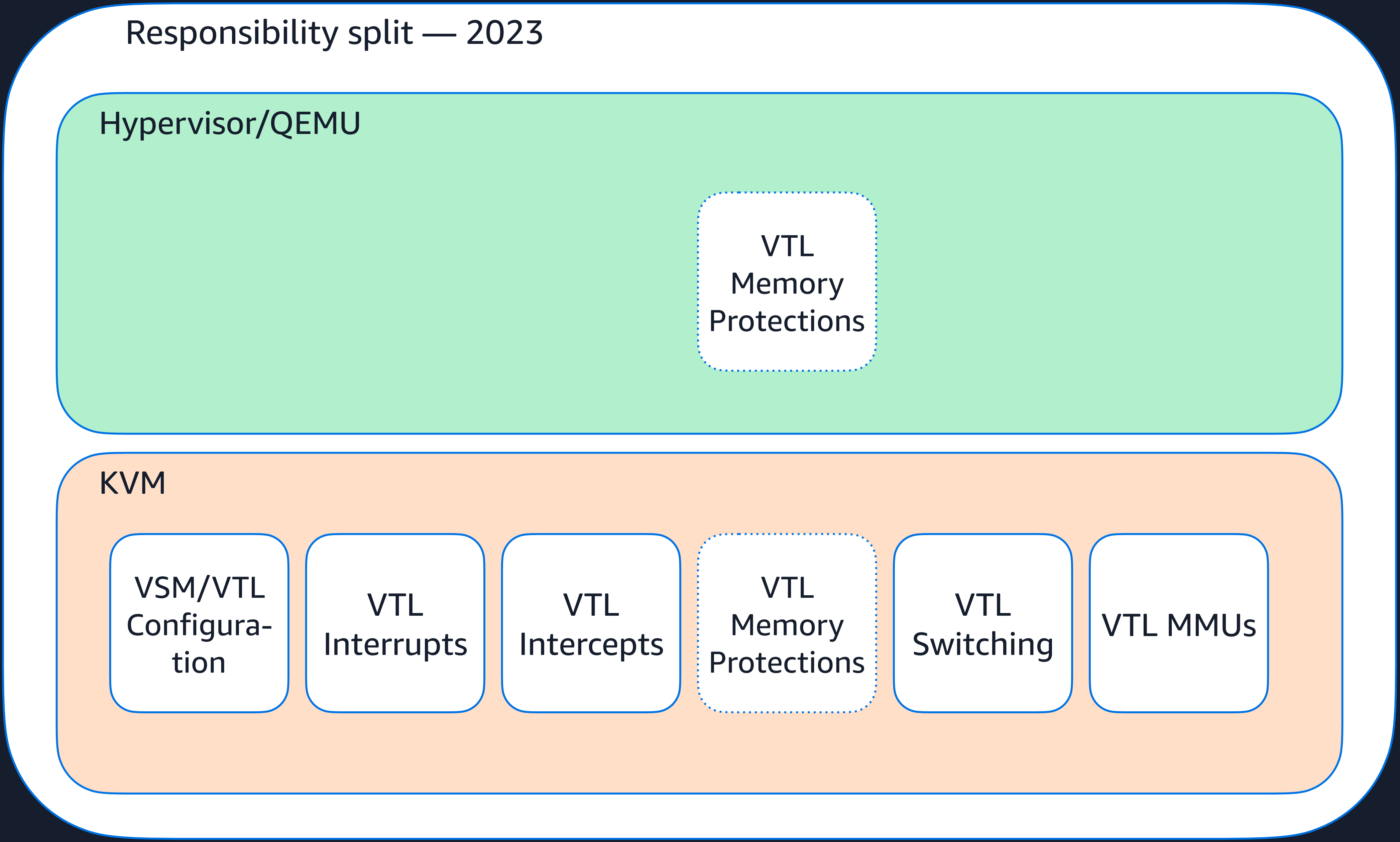
What about I/O?

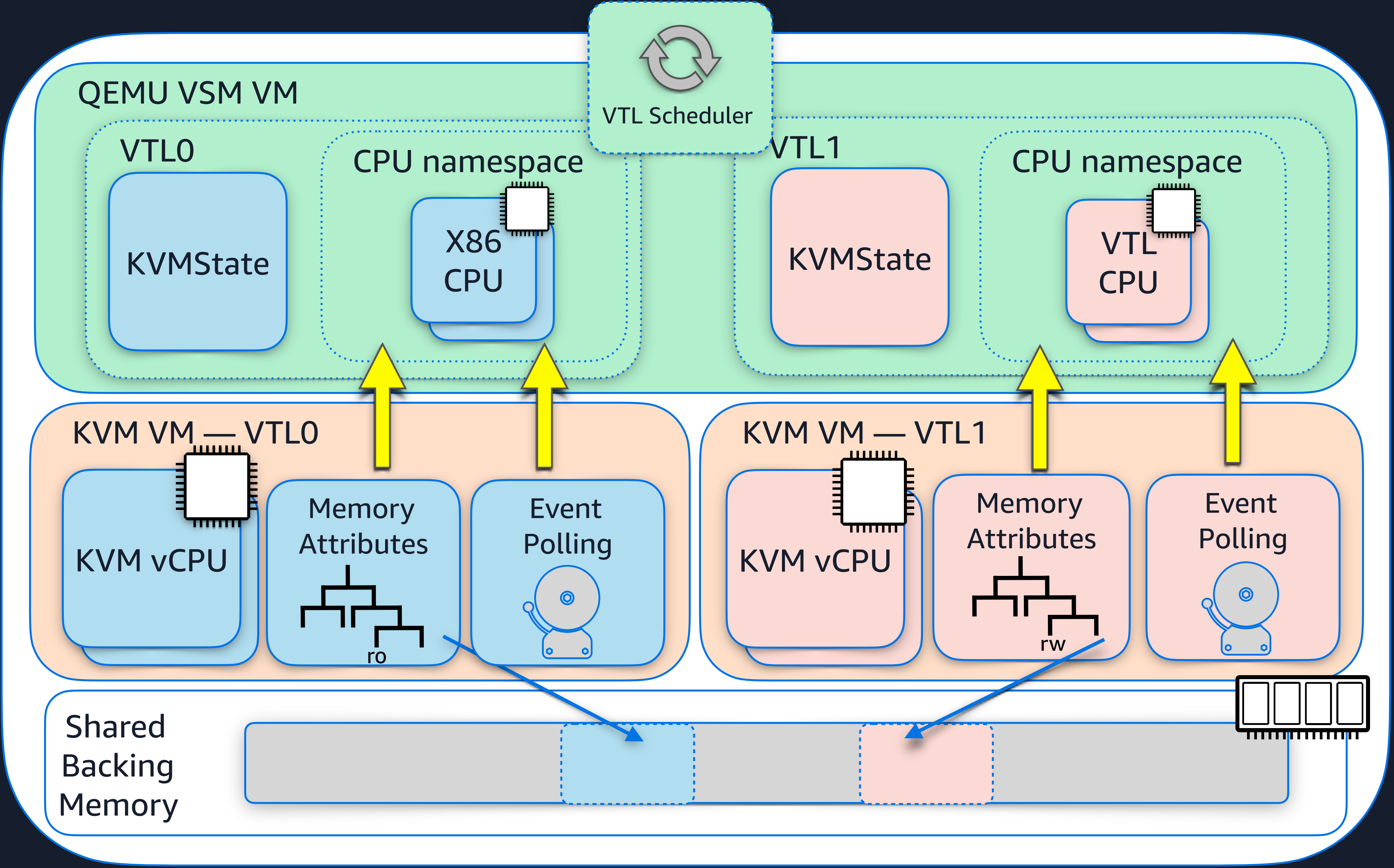

Host
VM
VTL0 — *Normal World*
Userspace
Kernel
Virtualized HW
VTL1 — *Secure World*
Userspace
Kernel
Virtualized HW
Hypervisor/QEMU
VSM
Linux Kernel
KVM
Hardware

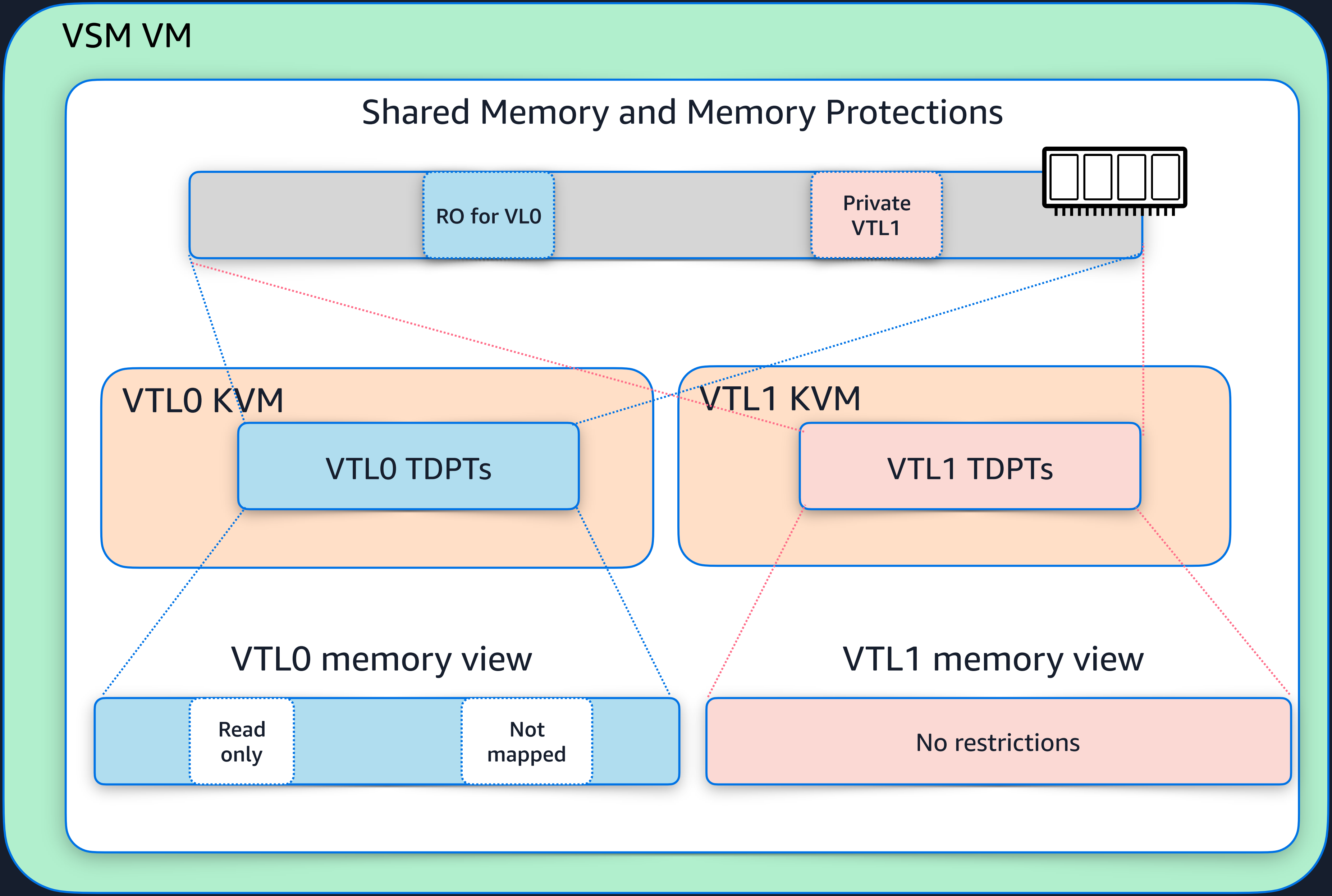

Memory
read-only
Hyper-
visor
VTL0
TDPs
ro
VTL switch
VTL1
TDPs
rw
VM
VTL0 — Normal World
User-space
Kernel !!!
Virtualized HW
!!!
VTL1 — Secure World
User-space
Kernel
Virtualized HW

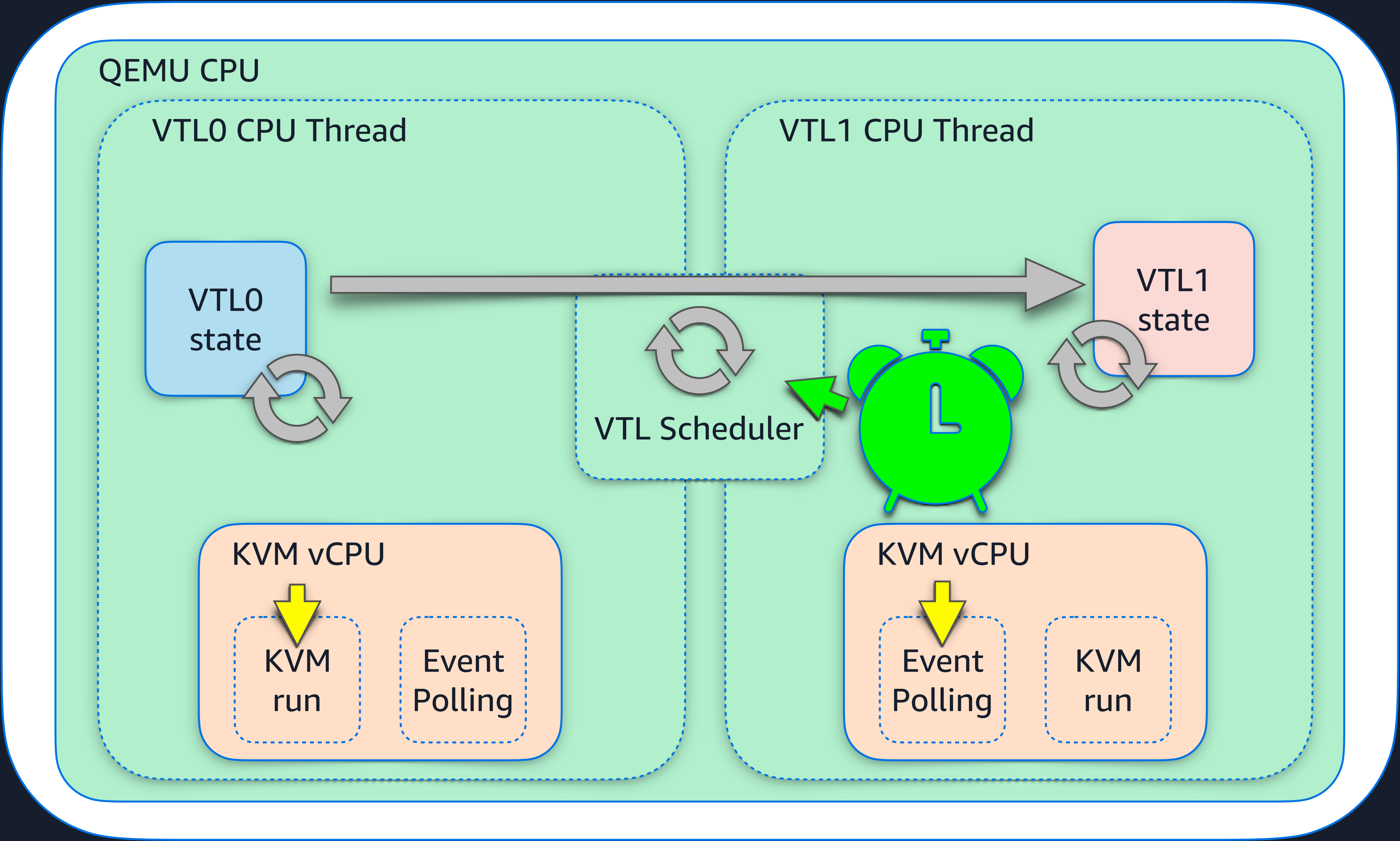
Where can I use this?

Emulating Hyper-V VSM In QEMU/KVM


Responsibility split — 2023


VTL Scheduler
QEMU VSM VM
VTL0
CPU namespace
KVMState
X86 CPU
VTL1
CPU namespace
KVMState
VTL CPU
KVM VM — VTL0
KVM vCPU
Memory Attributes
ro
Event Polling
KVM VM — VTL1
KVM vCPU
Memory Attributes
rw
Event Polling
Shared Backing Memory


VSM VM
Shared Memory and Memory Protections
RO for VL0
Private VTL1
VTL0 KVM
VTL0 TDPTs
VTL1 KVM
VTL1 TDPTs
VTL0 memory view
Read only
Not mapped
VTL1 memory view
No restrictions


QEMU CPU
VTL0 CPU Thread
VTL1 CPU Thread
VTL0 state
VTL1 state
VTL Scheduler
KVM vCPU
KVM run
Event Polling
KVM vCPU
Event Polling
KVM run

Responsibility split rethought

Hypervisor/QEMU

- VSM/VTL Configurati on
- VTL Interrupts
- VTL Intercepts
- VTL Memory Protections
- VTL Switching

KVM

- Hyper-V VSM hcall pass-through
- vCPU Event Polling
- CR/MSR filtering
- RWX Memory Attributes/ Fault Exits
- KVM Translate2/ TLB Flush Inhibit

WIP

Upstream status

Available as Linux/KVM series:

- Core Hyper-V VSM Enablement
- RWX Memory Attributes
- KVM Translate2
- HvTlbInhibit

Upcoming Linux/KVM RFC:

- CPU Register Filtering
- MBEC Aware Memory Attributes

Upcoming QEMU RFC:

-

When can I run VSM with QEMU/KVM?

VSM on Linux Guests?!

VTL2?
Para-visors?
Device emulation in guest context?

Thanks!

References

VSM development repositories:

- *https://github.com/vianpl/linux* `vsm/next`
- *https://github.com/vianpl/qemu* `vsm/next`
- *https://github.com/vianpl/kvm-unit-tests* `vsm/next`
- *https://github.com/vianpl/qemu-kvm-dev-env* *`master`*

Upstream Series:

- *https://lore.kernel.org/kvm/20240609154945.55332-1-nsaenz@amazon.com/*
- *https://lore.kernel.org/linux-doc/20240910152207.38974-1-nikwip@amazon.de/*

VBS in Linux: *https://lssna24.sched.com/event/1aIeD*

Contact: *nsaenz@amazon.com*

Per-VTL Private State

```
SYSENTER_CS, SYSENTER_ESP, SYSENTER_EIP, STAR,
LSTAR, CSTAR, SFMASK, EFER, PAT,
KERNEL_GSBASE, FS.BASE, GS.BASE, TSC_AUX
HV_X64_MSR_HYPERCALL, HV_X64_MSR_GUEST_OS_ID
HV_X64_MSR_REFERENCE_TSC,
HV_X64_MSR_APIC_FREQUENCY
HV_X64_MSR_EOI, HV_X64_MSR_ICR
HV_X64_MSR_TPR, HV_X64_MSR_APIC_ASSIST_PAGE
HV_X64_MSR_NPIEP_CONFIG
HV_X64_MSR_SIRBP, HV_X64_MSR_SCONTROL
HV_X64_MSR_SVERSION, HV_X64_MSR_SIEFP
HV_X64_MSR_SIMP, HV_X64_MSR_EOM
HV_X64_MSR_SINT0 – HV_X64_MSR_SINT15
HV_X64_MSR_STIMER0_CONFIG –
HV_X64_MSR_STIMER3_CONFIG
HV_X64_MSR_STIMER0_COUNT –
HV_X64_MSR_STIMER3_COUNT

Local APIC registers (including CR8/TPR)
RIP, RSP, RFLAGS
CR0, CR3, CR4
DR7, IDTR, GDTR, CS, DS, ES, FS, GS, SS,
TR, LDTR, TSC
```

Per-VTL Shared State

```
HV_X64_MSR_TSC_FREQUENCY
HV_X64_MSR_VP_INDEX
HV_X64_MSR_VP_RUNTIME
HV_X64_MSR_RESET
HV_X64_MSR_TIME_REF_COUNT
HV_X64_MSR_GUEST_IDLE
HV_X64_MSR_DEBUG_DEVICE_OPTIONS
HV_X64_MSR_BELOW_1MB_PAGE
HV_X64_MSR_STATS_PARTITION_RETAIL_PAGE
HV_X64_MSR_STATS_VP_RETAIL_PAGE
MTRRs
MCG_CAP
MCG_STATUS
Rax, Rbx, Rcx, Rdx, Rsi, Rdi, Rbp
CR2
R8 – R15
DR0 – DR5
X87 floating point state
XMM state
AVX state
XCR0 (XFE)
```