Nesting Secure Hosts

Secure VMs in nested hypervisors

—

Janosch Frank
frankja@de.ibm.com

IBM

Legal

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or TM), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

The following are trademarks or registered trademarks of other companies.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notes

- My experiences with this topic are s390x / IBM Z centric

➔ **If you want to know if something can be implemented in architectures other than s390, then please reach out to the respective developers and maintainers**

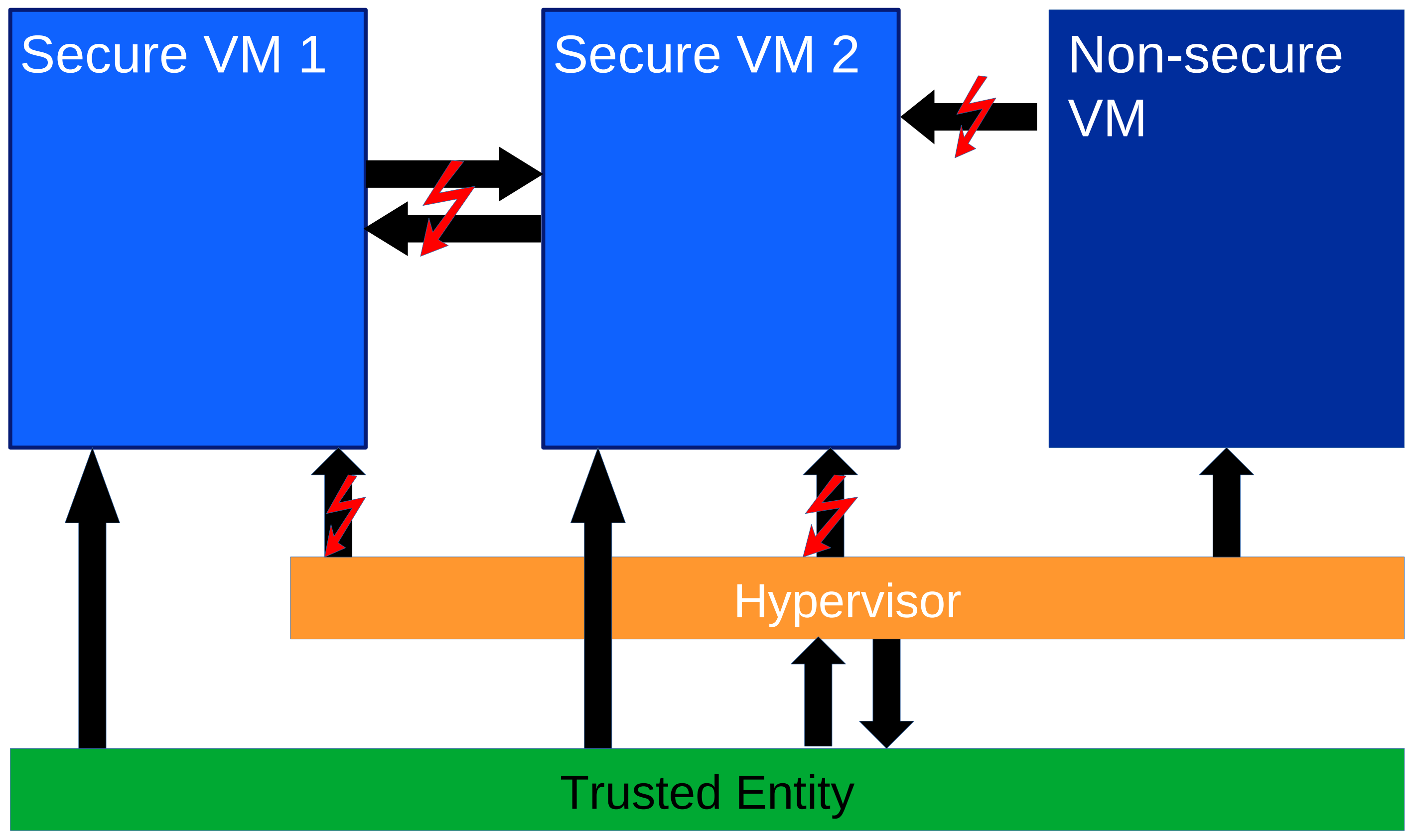
Contents

Introduction

Recap – Secure VMs

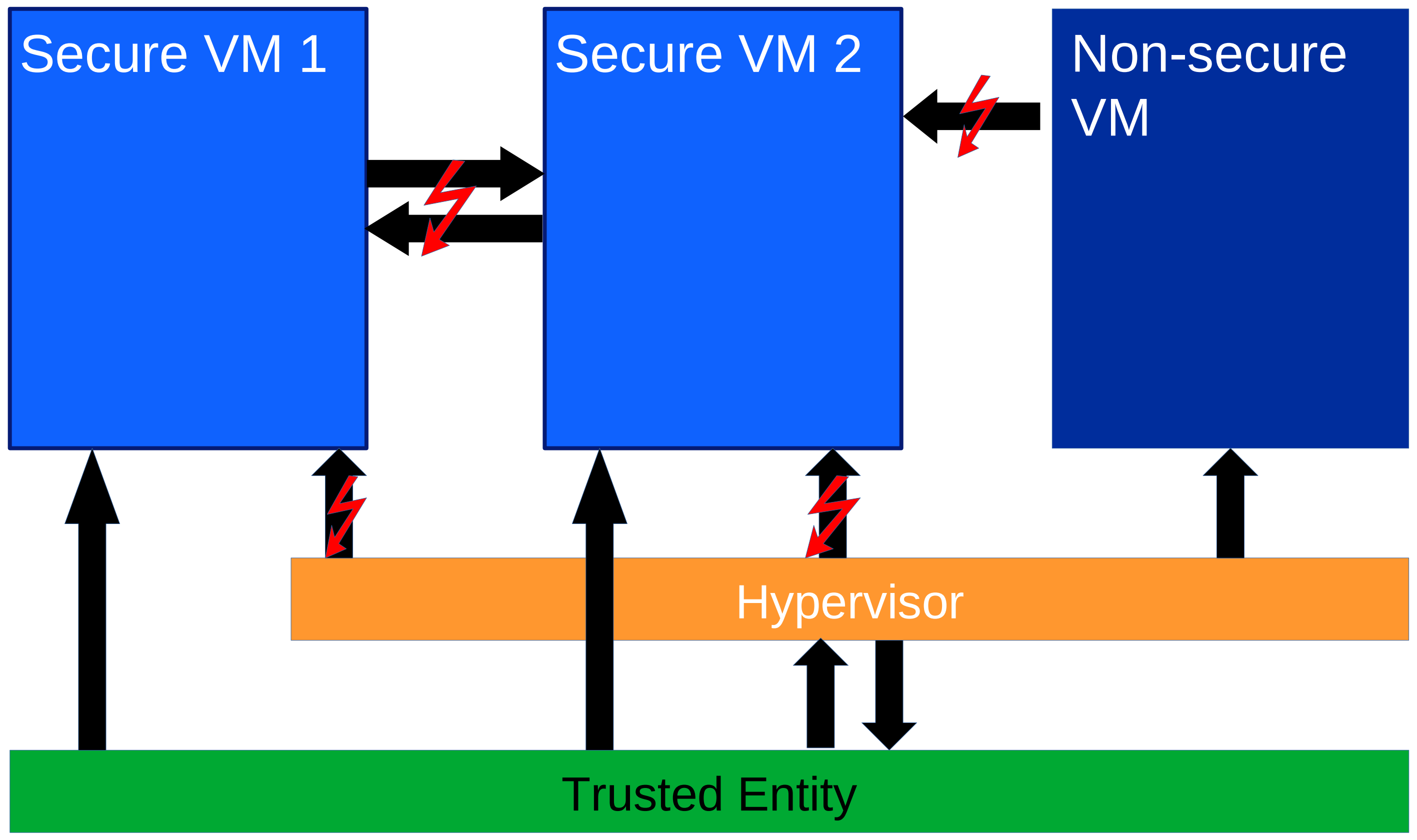
- VM sensitive state: Registers, memory, IRQs, …
- Secure VM’s sensitive state is not accessible from the OS / hypervisor
- Confidentiality and often also integrity protection

- A Trusted Entity manages sensitive VM data
- Hypervisor cooperates with the Trusted Entity to run secure VMs
- Confidential VM == Secure VM



Recap – Secure VMs

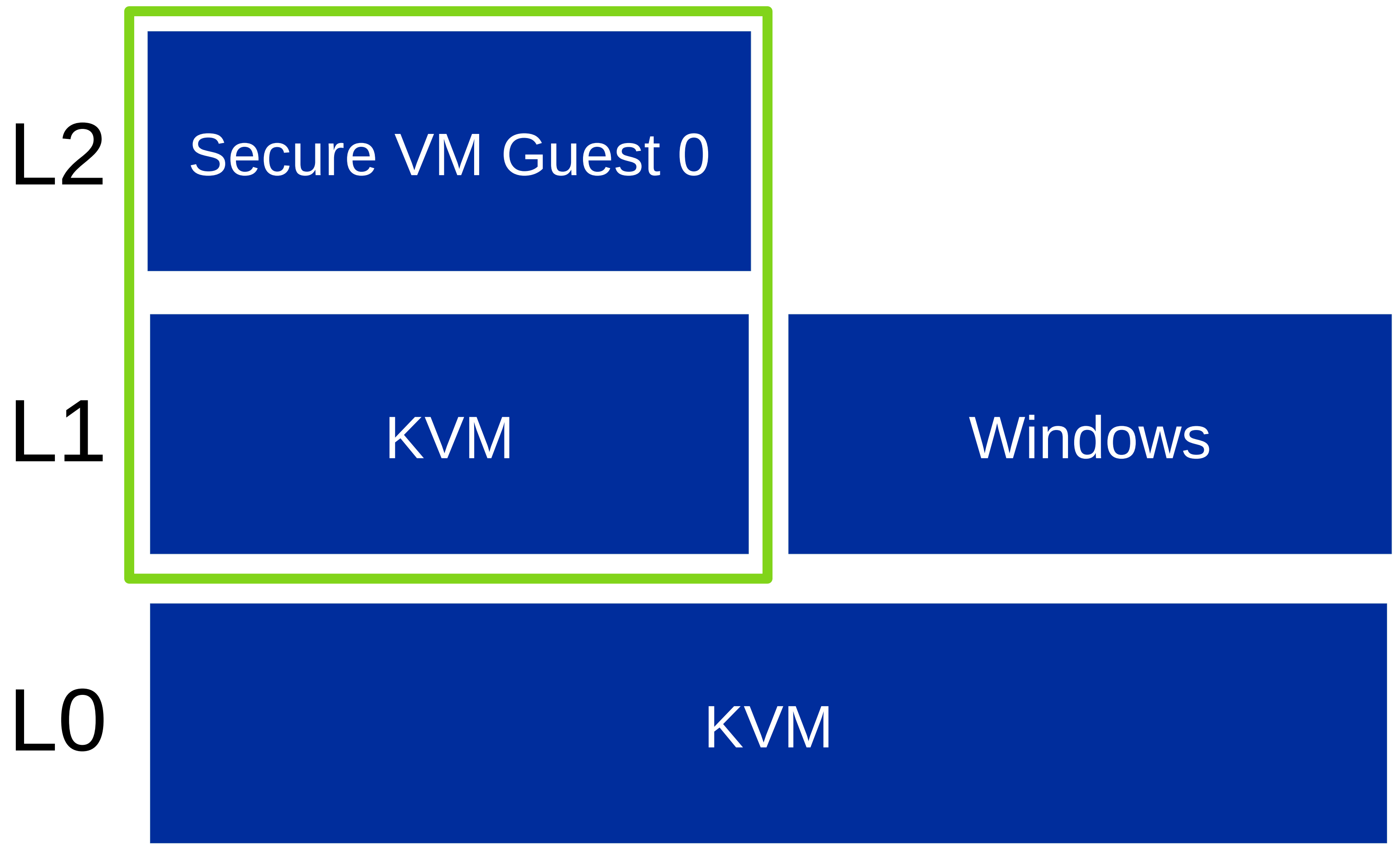
- AMD SEV
- Intel® TDX
- IBM® Z Secure Execution
- IBM® POWER Protected Execution



What are we talking about?

A KVM VM being a host for a secure VM.

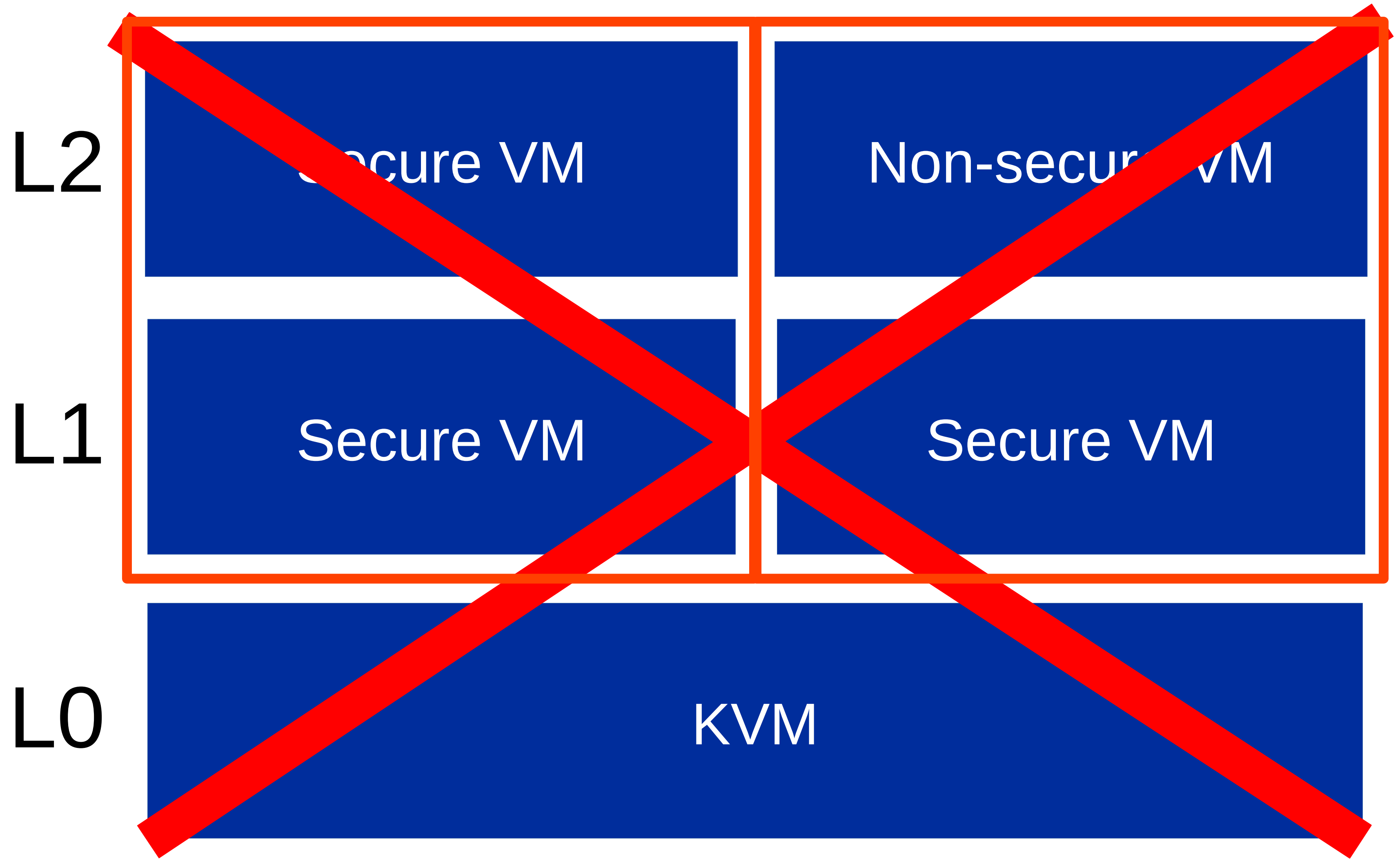
A secure VM being run as a nested guest.



What are we NOT talking about?

We’re not talking about a secure VM being a VM host to:

- Non-secure VM
- Secure VM



What are we NOT talking about?

Nesting works by reading and modifying guest memory in order to emulate virtualization.

But secure VMs don’t allow memory access:

- Nesting secure VMs would only be possible with hardware and Trusted Entity support
- Adding that native nesting support would mean a **massive development effort**

Why?

Use cases are not different from non-secure use cases:

- Hypervisor & test development
- Moving complete computing environments into the cloud
  - Making KVM hosts manageable without hardware access
  - Usage of VM or container management tools

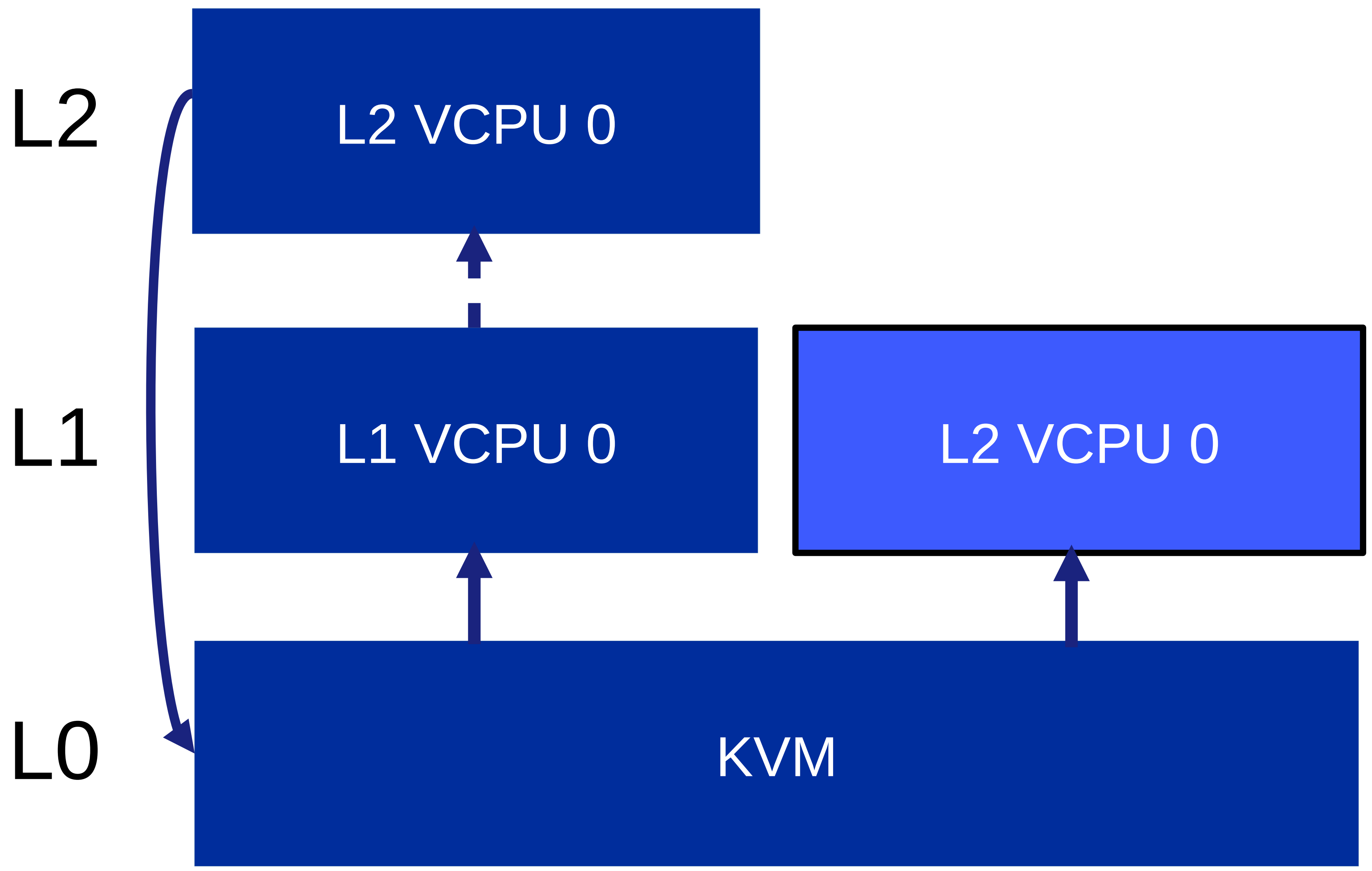
How

Nesting

- Three parts
  - Secure VCPU nesting
  - Trusted Entity ABI emulation
  - Architecture compliance

Nesting - VCPU

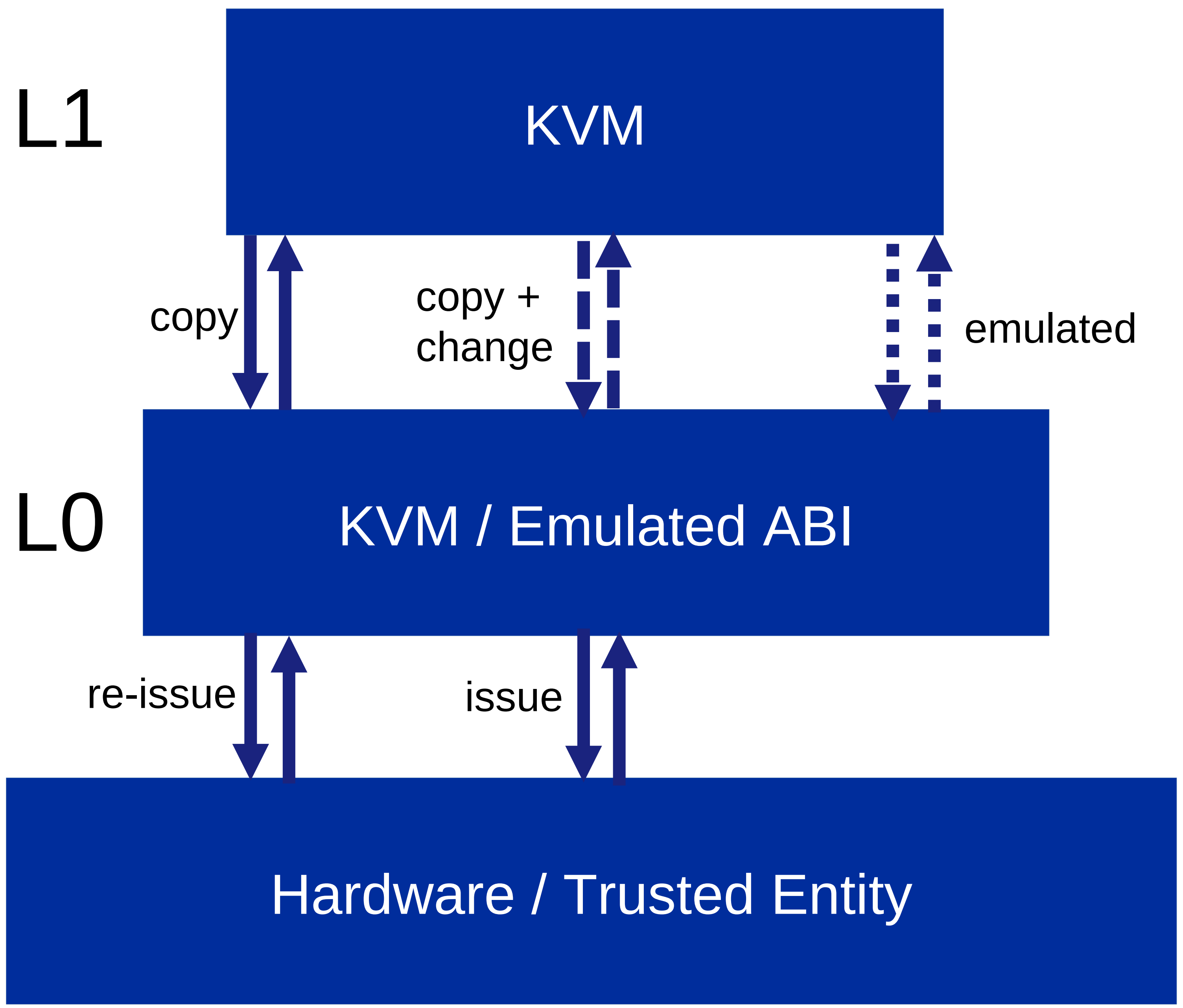
- Running a nested VCPU is a solved problem
  1) The Level 1 VCPU exits to Level 0 KVM when Level 1 tries to run a nested VCPU
  2) KVM shadows the nested VCPU control structures
  3) The shadow control structure is used to run the nested Level 2 VCPU as a normal Level 1 VCPU
- Nesting secure VCPUs is relatively easy if that concept can be maintained



Nesting - ABI emulation

- Emulation of the Level 1 ABI is done by re-using the Level 0 Trusted Entity ABI
- Level 0 KVM tracks secure entities for each Level 1 secure host
- Level 1 ABI requests are then modified so the Level 0 Trusted Entity understands them

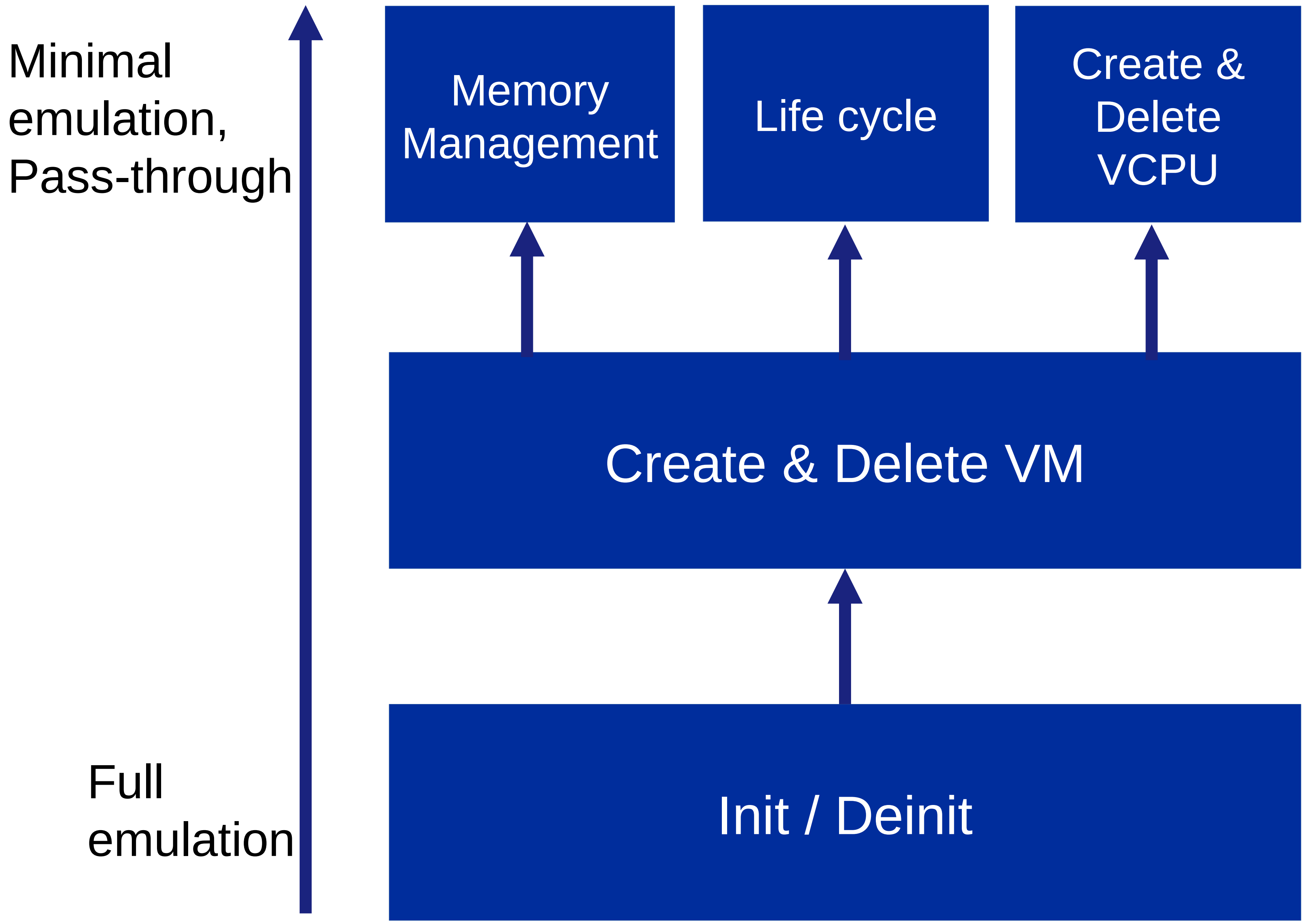
- Some ABI calls can potentially be passed through
- Most are executed with heavily modified data
- A small amount is fully emulated



Nesting - ABI emulation

- Trusted Entity calls can be sorted into the following categories:
  - Initialization & teardown of the Trusted Entity
  - VM management
  - VCPU management
  - VM memory management
  - VM life cycle calls (Dump, resets, IRQs)

  - **The interface is hierarchical**
  - **We tend to need less emulation the higher we go**

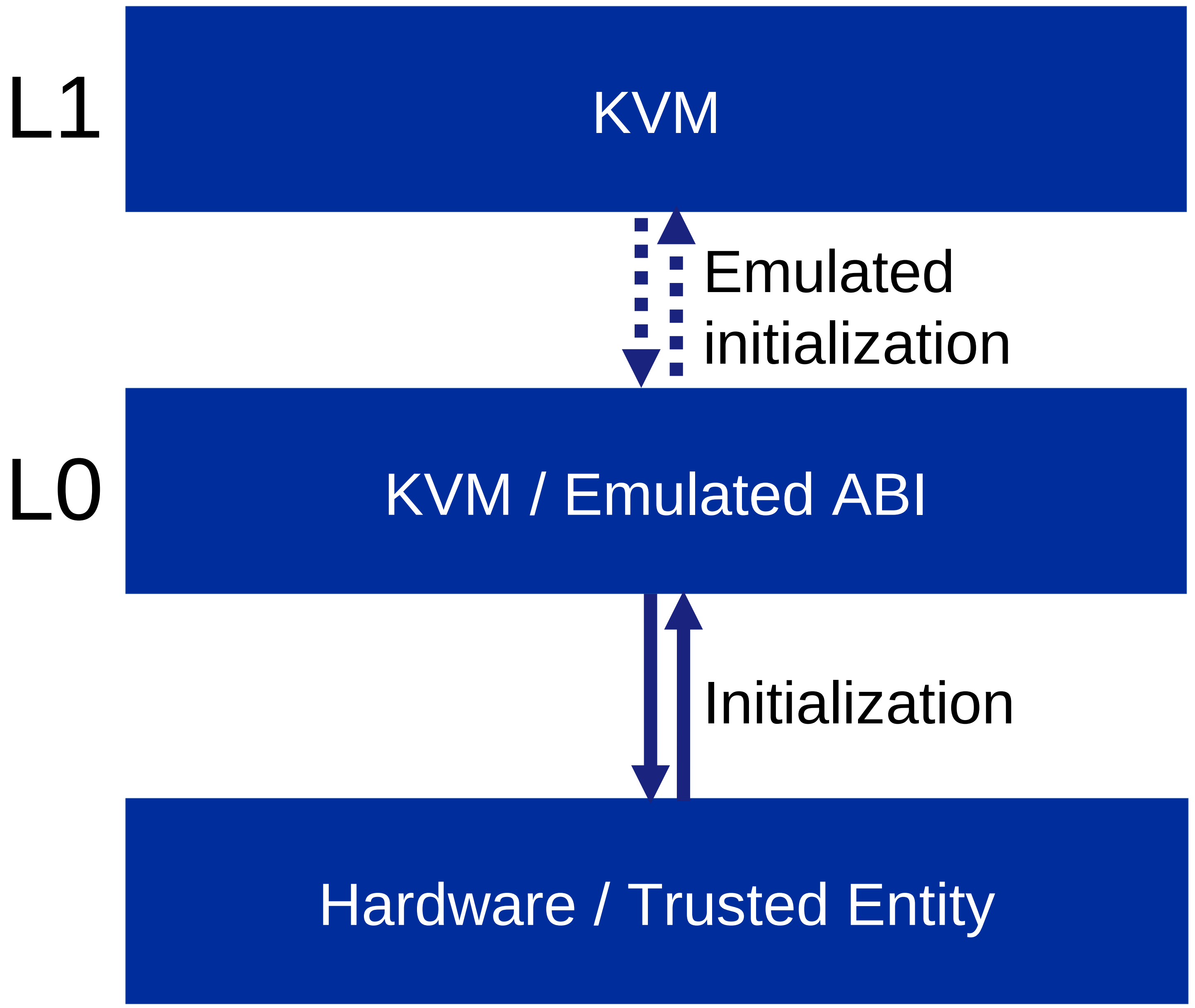


*Approximation of interface

Emulation

Trusted Entity ABI emulation – Host environment

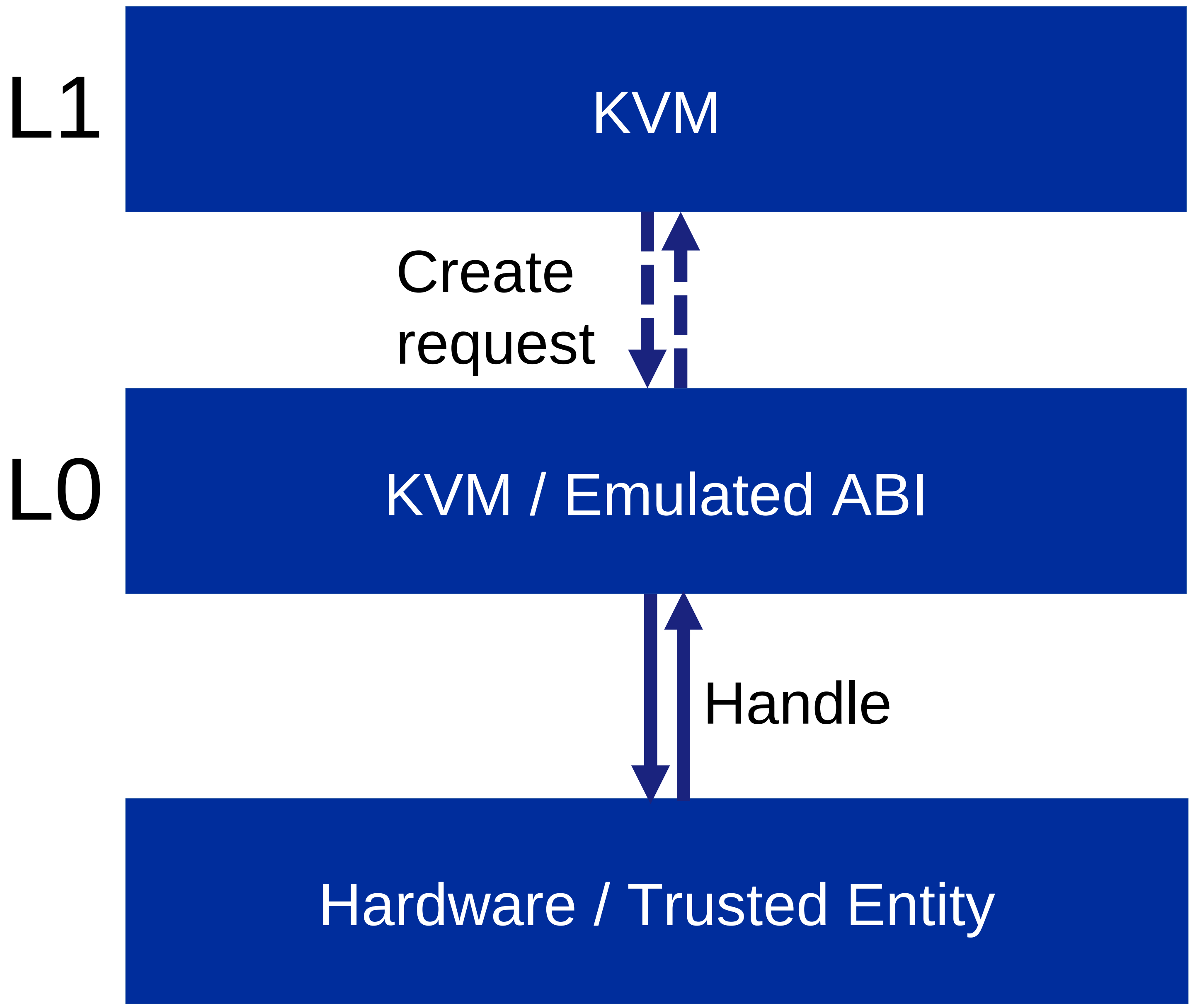
- Trusted Entity has to be initialized in order to create and manage secure VMs
  - S390 Initalize Ultravisor call
  - AMD SEV INIT & SHUTDOWN commands
  - Intel® TDX TDH.SYS.INIT & TDX TDH.SYS.SHUTDOWN
- Level 0 initializes TE on boot or when KVM is loaded
- Therefore these ABI calls will likely be fully emulated for Level 1
- Further configuration of the TE might be a major hurdle to emulation (FW update, key management)



Trusted Entity ABI emulation – VM management

- Secure VMs and their VCPUs are created via ABI calls
- The Trusted Entity will return a handle (s390x, AMD) or an address is used as handle (Intel®)
- The handle is the identifier of the secure VM / VCPU for all following ABI calls

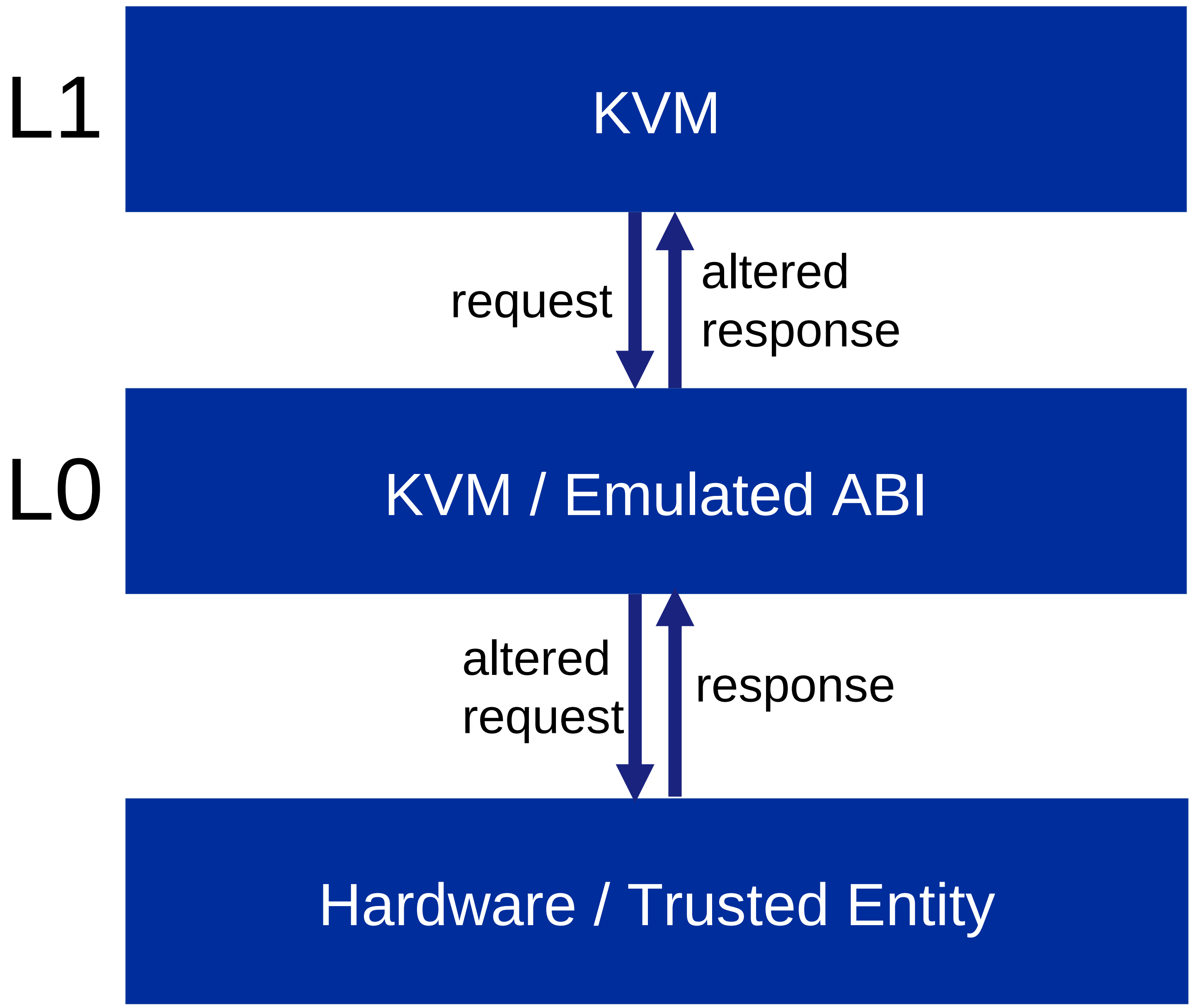
- These calls are re-issued with manipulated data for emulation
- Passing the handles on has its benefits
  - Calls with only the handle as input can likely be passed through unmodified
  - No mapping of emulated handles to real handles



Trusted Entity ABI emulation – Lifecycle calls

- Examples:
  - VCPU state registration (running, stopped)
  - Resets of VM state
  - Dump
  - Migration

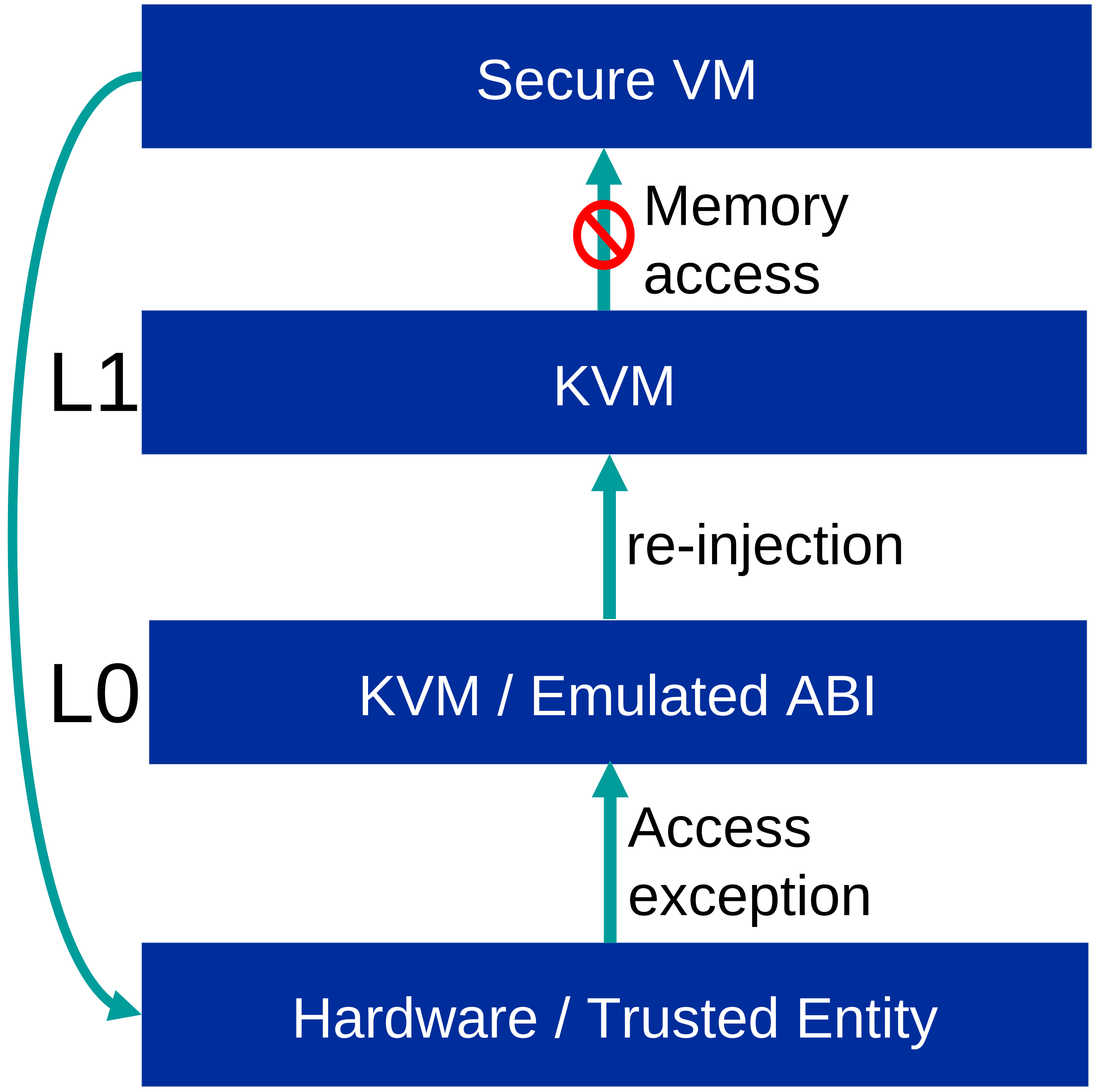
- Most of these are pretty easy to handle
- Especially if the data is only handle + command
- If addresses are involved they will have to be translated or bounce buffered



Architecture Compliance

Architecture Compliance – memory protection

- Secure VM memory is protected against all external accesses
- Protection can be provided via access exceptions
- Depending on architecture, we may need to re-inject access exception into the KVM host



Architecture Compliance – TE structure access

- Some architectures define specific memory areas as owned by the Trusted Entity
- It stores management data for secure VMs or its global state in there
- Once Trusted Entity takes ownership of a memory area any OS access is denied

➔ To fully emulate the architecture KVM needs to restrict access to emulated trusted entity structures

➔ Which means tracking them and removing the possibility of access

Architecture Compliance – TE structure access

- Donations could be disabled entirely
- But then there’s no way for the emulated ABI to re-gain the memory it needs to donate to the real Trusted Entity in order to emulate ABI calls
- Also those donations limit the amount of secure resources that can be created

Problems

Required virtualization features

- Secure VMs often require the usage of virtualization enhancement facilities
- Those facilities have initially been introduced to speed up virtualization
- But they also provide security because handling of sensitive things moves into Trusted Entity

- Examples:
  - Trusted Entity IRQ injection (exceptions, IO, signaling)
  - Trusted Entity handling of complex instructions

Required virtualization features

- Secure VM technology is a recent addition to the architecture
- No need (yet) to fence features
- Therefore enabling as many architecture features as possible made sense

But:

- ➔ Not all of those features are compatible with nesting
- ➔ This can’t be resolved without hardware and / or Trusted Entity changes

Page management

- Secure VMs have page integrity and (re-)map protection
- Page meta data is stored in some kind of table
- Trusted Entity maintains the table
- Hypervisors can request page mapping changes from Trusted Entity, e.g. to map a new page

➔ Every change request is emulated

➔ Each additional exit costs performance

Page management

- Fault driven secure memory management allows a few workarounds
- Pre-fault:
  - Most of the page map requests are processed at the start of a secure VM
  - With a bit of hypervisor code we could fault in and secure all memory in one go with a small number of exits
  - The two faults could be merged into one
    - Either by faulting automatically when making a page secure
    - Or by making a page secure when faulting it in

Migration

- Migration of a secure VM KVM host will be challenging
- Easier to re-create the nested KVM host and then migrate the secure VMs to it instead
- Re-creation would need to be implemented

➔ For now disabling migration of the host VM is more likely

Reboots

- Emulated Trusted Entity might need to be torn down on secure host reboot
- Need to catch sudden reboots of secure host VM
- ABI teardown interface can not be used

- ➔ Initialization and deinitialization are fully emulated
- ➔ Need to destroy all secure VCPUs and VMs via real Trusted Entity
- ➔ Can affect reboot times significantly

Reboots

- If an architecture uses memory donation there’s an additional step
- The access protection for every page of emulated donated storage will also need to be removed

Implementations

Implementations

- S390 has done experiments
  - S390 Trusted Entity ABI is small (currently 28 calls, some are guest only)
  - POC runs Linux secure guests and most KVM unit tests are green
  - It is not fully architecture compliant
  - Two weeks of initial work until Linux was booting / stable

- It will be a **lot** of code and testing
  - Lots of Trusted Entity ABI error checks
  - Each read or write accessing the nested secure VM host needs access exception handling
  - Lots of code to test

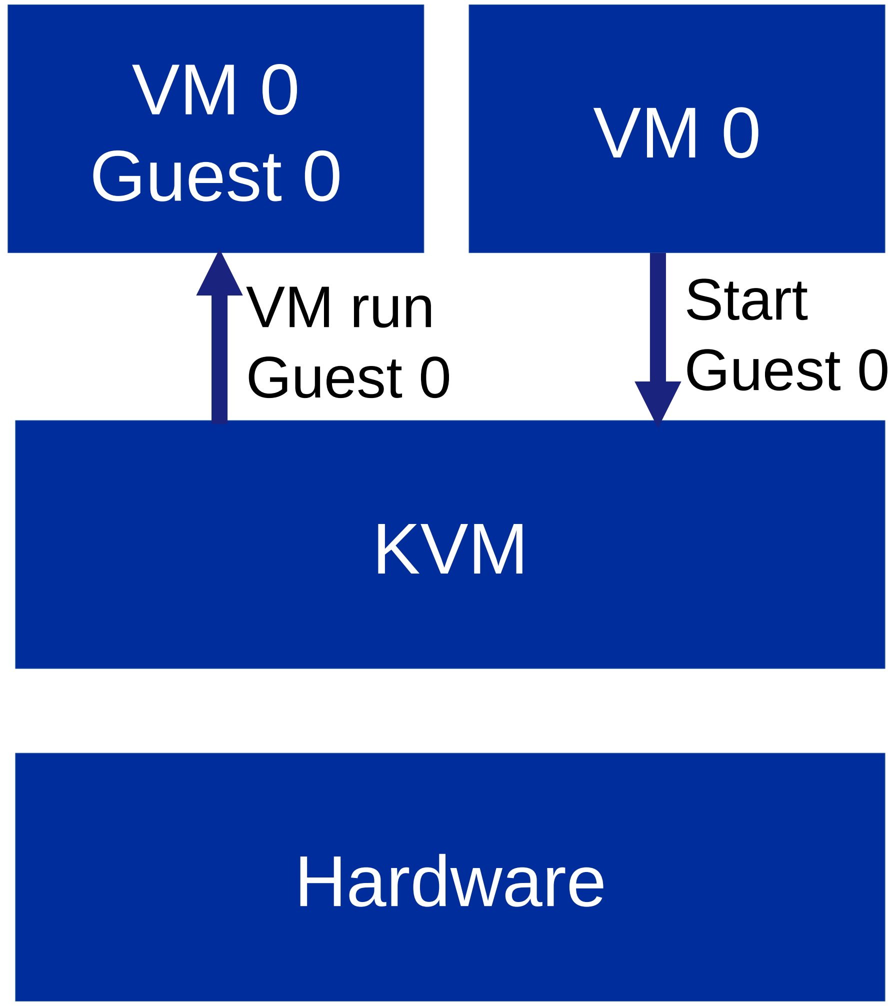
Implementations

S390 Proof of concept:

- 2k lines for KVM
  - 80% interface emulation
  - 10% memory management
  - 10% secure VCPU nesting
- Minimal QEMU changes
  - Feature enablement and indication
  - Migration blocker

Implementations

- Other solutions are providing a virtualization hosting interface
- A VM can request the host to start a VM (VSOCK, network, ...)
- The started VM is a de facto guest of the requesting VM



- ✓ Faster
- ✓ Easier access to host hardware
- ✓ Potentially only a userspace change
- ✗ Less flexible
- ✗ More host tracking

Addressing potential questions

- I wasn't able to measure the performance impact yet
- I haven't even thought about multilevel nesting
- The migration statement is a theoretical one as s390 doesn't support migration right now
- Is this possible on architecture xyz?
  - My current **guess** is that AMD SEV is the next best candidate
  - Encryption vs access protection shouldn't matter

Thank you

Janosch Frank

—

frankja@de.ibm.com

IRC: frankja @ #kvm #zkvm oftc.net

IBM