Virtualization from an attacker point-of-view

An introduction to VM escapes

New is not always better.

GREHACK

11

Who are we

Thomas BOUZERAR

SECURITY EXPERT
@MajorTomSec

Corentin BAYET

SECURITY EXPERT
@OnlyTheDuck

SYNACKTIV

- Offensive security
- 170 Experts
- Pentest, reverse engineering, development, incident response

Reverse Engineering team

- 45 reversers
- Low level research, reverse engineering, vulnerability research, exploit development, etc.

Introduction

About this talk

What we WILL talk about

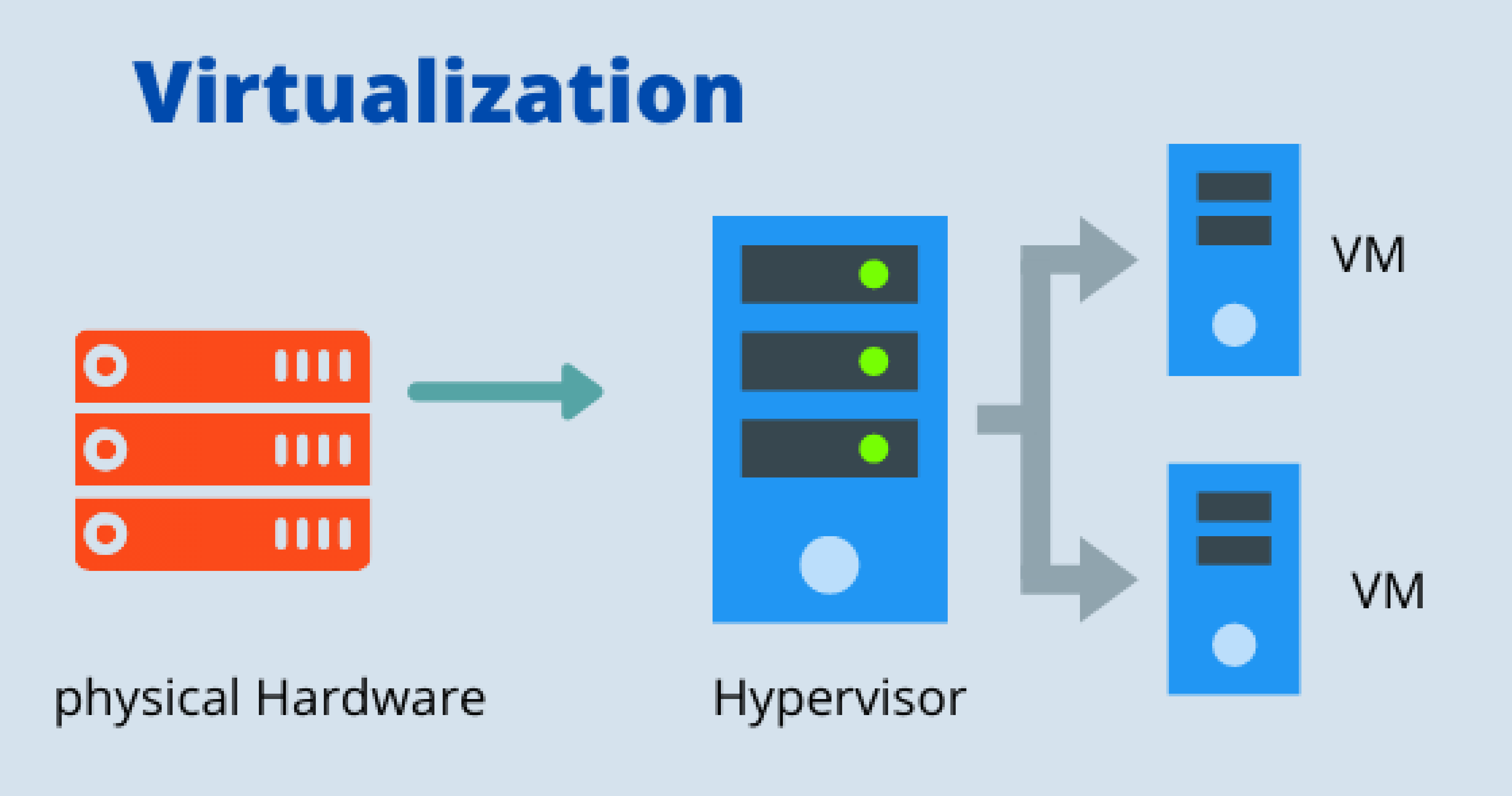
- What is virtualization and how it works
- The attack surface exposed by an hypervisor
- History of bugs found in various components

What we WON’T talk about

- Deep technical details on the implementation of virtualization
- Exploitation of the bugs

What is virtualization

A few definitions



https://multifunction.gr/en/virtualization/

- Virtualization creates the illusion of multiple (virtual) machines on the same physical hardware.
- The "host" software is called the hypervisor
  - Hyper-V, Xen, VirtualBox, VMware Workstation
- A Virtual Machine Monitor (VMM) is a part of the hypervisor that manages CPU, memory, I/O devices and interrupts
- The "guest" is the operating system which is running inside the virtual machine

What is virtualization

Role and objectives of an hypervisor

CPU and memory virtualization

Execute the instructions of the virtual machine in its own adress space.

Platform virtualization

Handle timers, interrupts, CPU traps…

IO devices virtualization

Emulate buses, graphics, network, disk…

Fidelity

Programs running in a virtual environment run identically to running natively.

Performance

The majority of guest instructions are executed by the hardware without the intervention of the VMM.

Safety

Resources are isolated between virtual machines and the host remains isolated from the guests.

What is virtualization

Role and objectives of an hypervisor

CPU and memory virtualization

Execute the instructions of the virtual machine in its own adress space.

Platform virtualization

Handle timers, interrupts, CPU traps…

IO devices virtualization

Emulate buses, graphics, network, disk…

Fidelity

Programs running in a virtual environment run identically to running natively.

Performance

The majority of guest instructions are executed by the hardware without the intervention of the VMM.

Safety

Resources are isolated between virtual machines and the host remains isolated from the guests.

Why ?

- The goal for an attacker is to escape a virtual machine and gain control of the hypervisor
  - VM escape (or VME)

- Very powerful primitive that can be critical for industries
  - Think about cloud computing or hosting that use virtualization

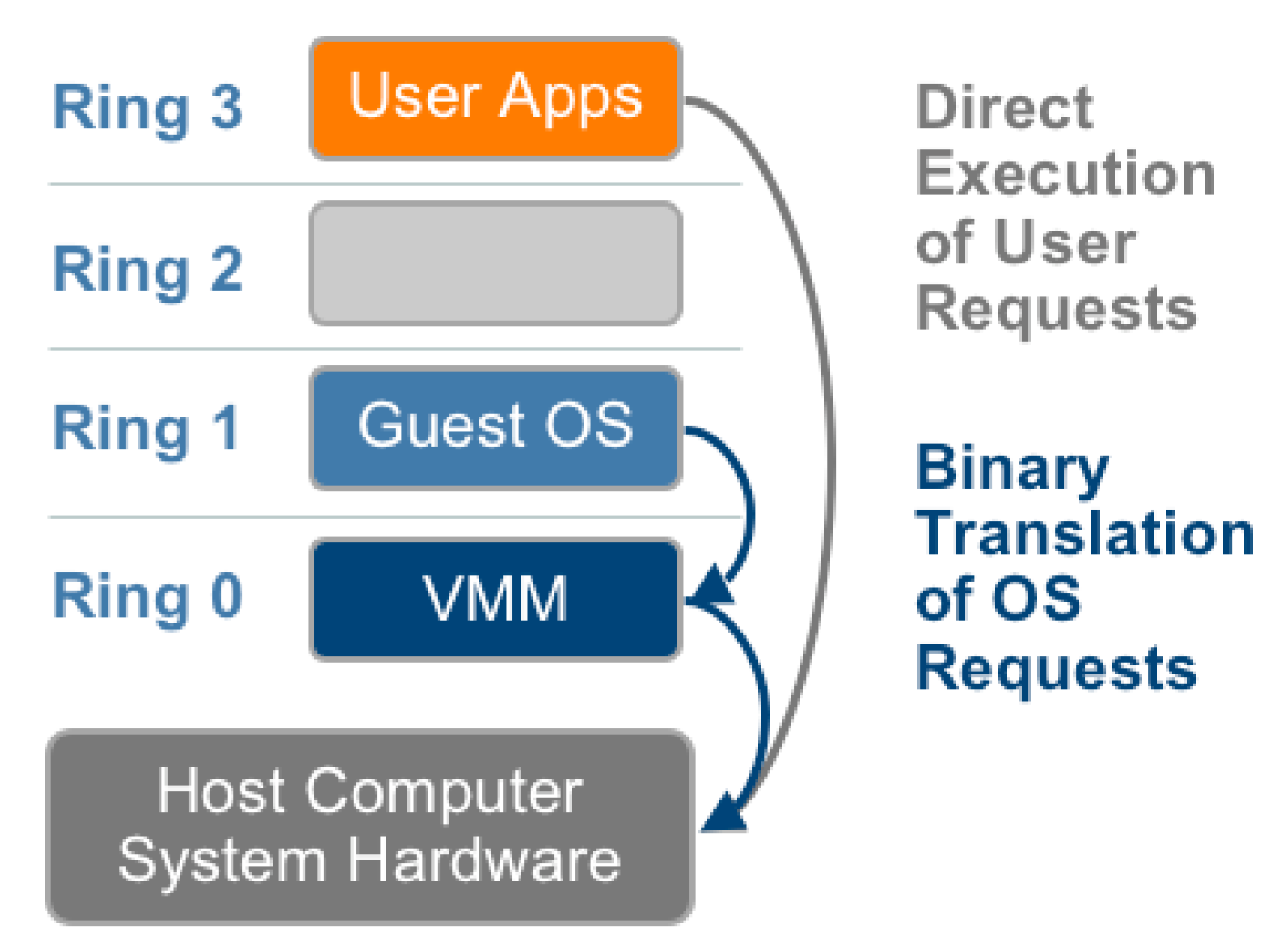
- **It's fun !**
  - By learning how virtualization works, you can learn how a computer actually works
    - But only by analyzing and reversing software !
    - … and reading intel’s manuals
- **It’s complex**
  - Very low level
  - Complex vulnerabilities and exploits

Virtualization basics

Virtualization techniques

Full virtualization with binary translation

Binary translation

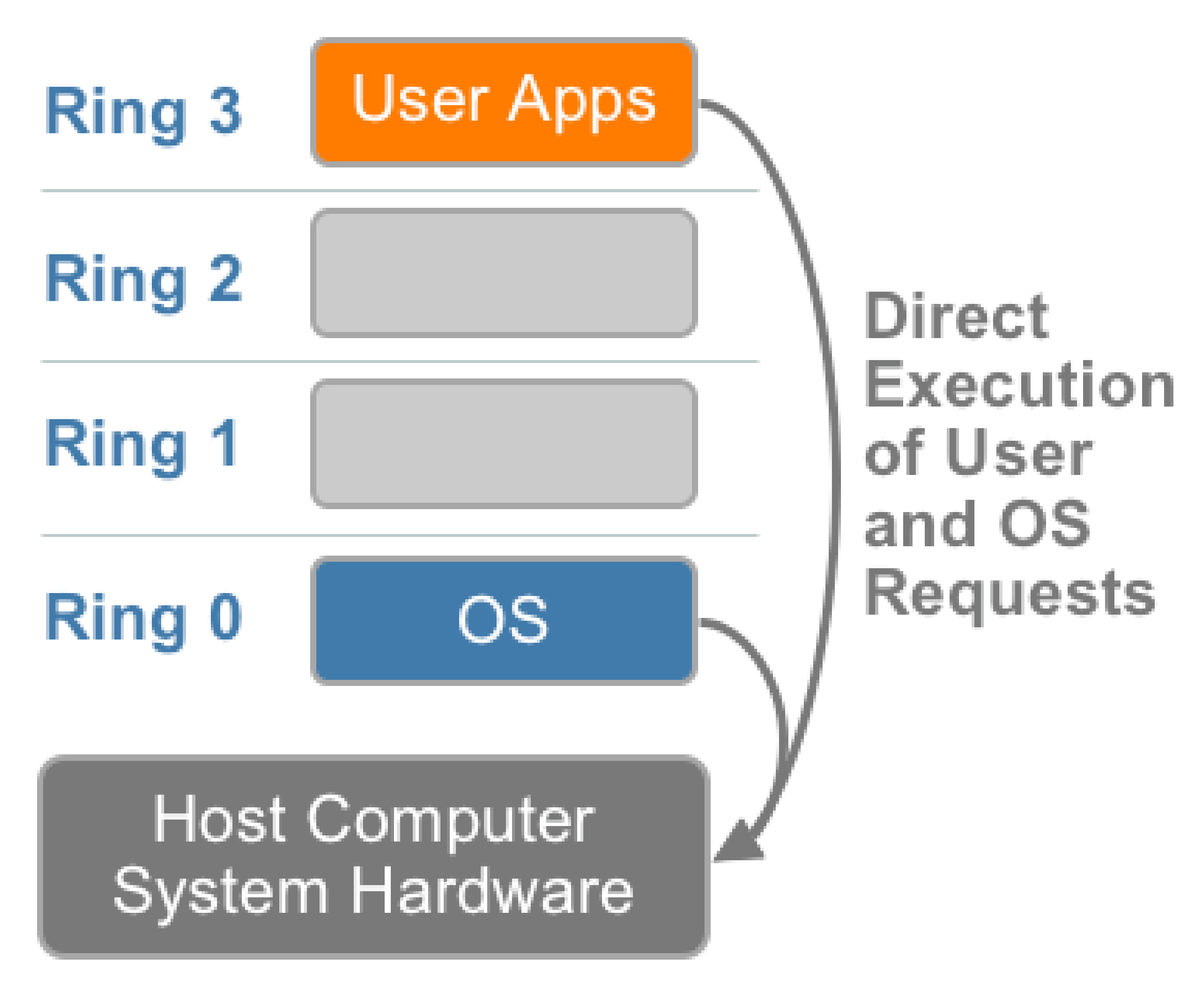


- First approach chosen by VMware for the first x86 full virtualization
  - Unprivileged instructions are executed directly on the CPU
- Guest's privileged instructions (I/O, interruptions…) are translated to traps and handled by the VMM
  - "Trap and emulate"
- Pros:
  - Guest OS has no idea that it is being virtualized
  - Good portability
- Cons:
  - A lot of CPU overhead for privileged instructions
  - Numerous traps
  - Very complex VMM

Virtualization techniques

Hardware assisted virtualization

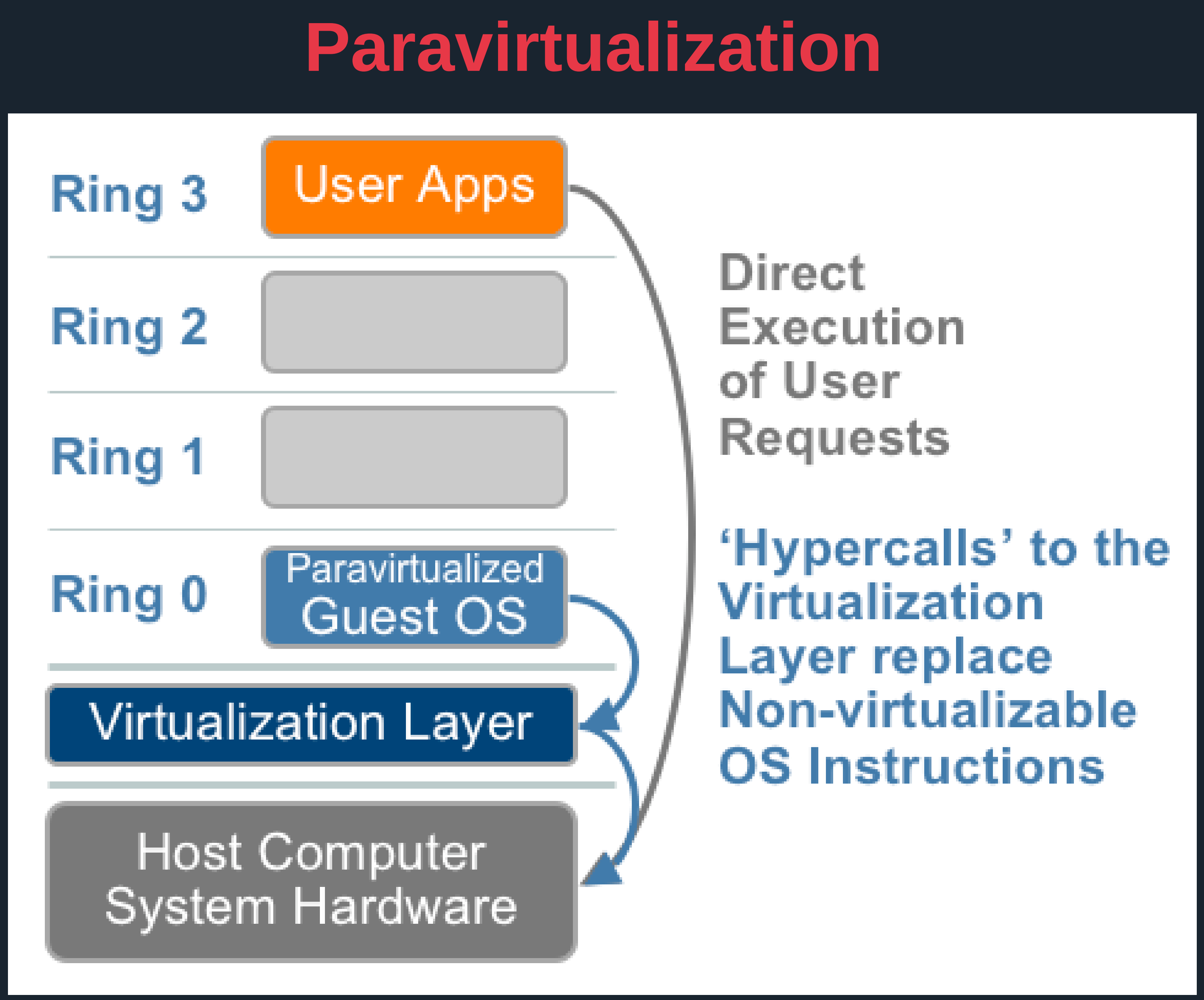
Hardware assisted



- CPU vendors started developing hardware features for virtualization
  - VT-X for Intel
  - AMD-V for AMD
- Hardware features for helping emulate the guest hardware
  - When the guest performs ”privileged operations”:
    - can be directly handled by the hardware (passthrough)
    - can give execution to the hypervisor (VMExit)
  - Avoids trapping all the time
    - Huge performance gain
- Not available on all CPUs !
  - But available on all modern ones
  - Most modern hypervisors require this feature

Virtualization techniques

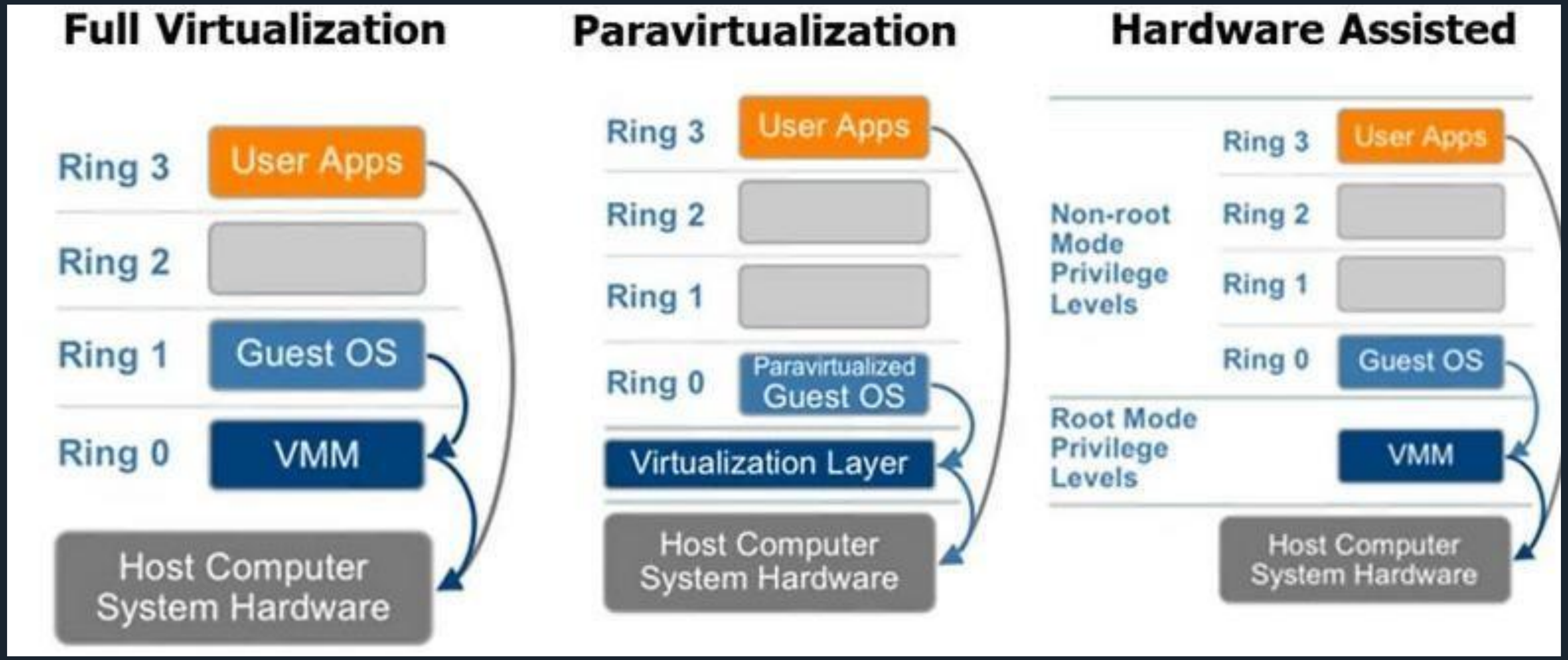
Paravirtualization



- Approach developed by Xen to have better performance
- An interface is developed on the guest to directly call the VMM and do privileged operations
  - "Hypercall"
  - Implements page faults, context switch, I/O operations...
- Pros:
  - Fast !
  - Simple VMM
- Cons:
  - Has to modify the guest
    - Not portable: has to develop an interface for every guest's kernel

Virtualization techniques

Architecture comparaison

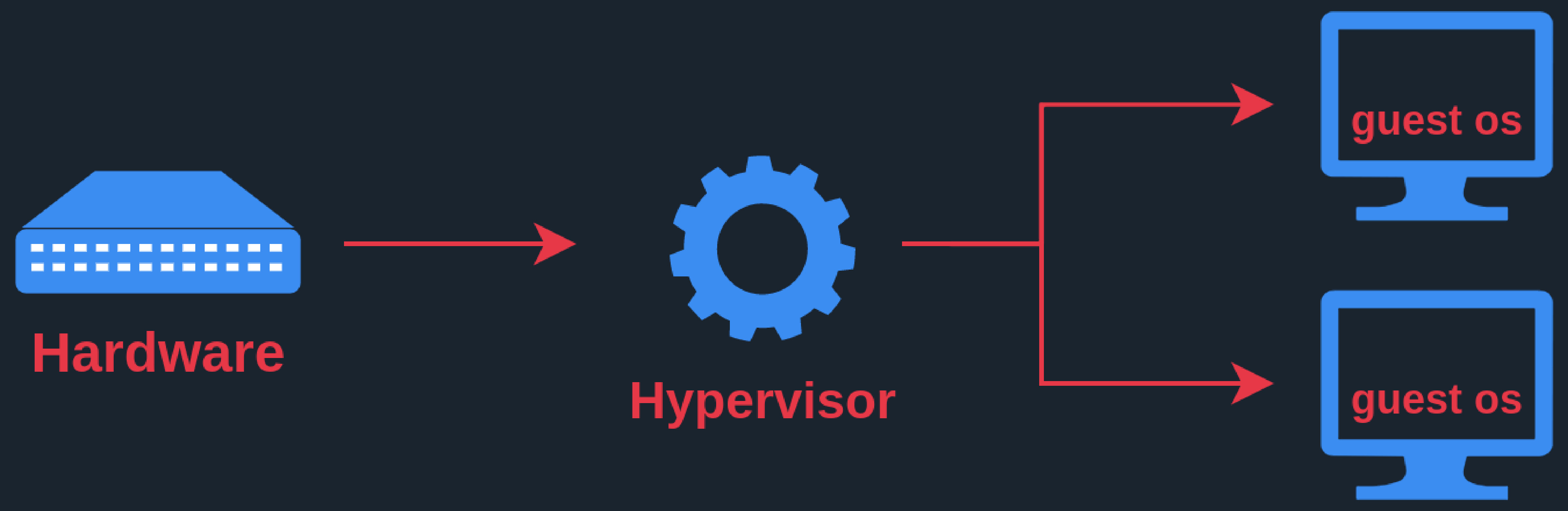


- Today, most hypervisors require hardware assisted virtualization to run
  - For CPU (VT-X, AMD-V)
  - For MMU (SLAT)
- “Trap and emulate” still used for complex privileged operations
  - Using VMENTER, VMEXIT
- Paravirtualization when stealthy virtualization is not necessary
  - For costing operations
  - For devices (network cards, …)

Software architecture

Type of hypervisors

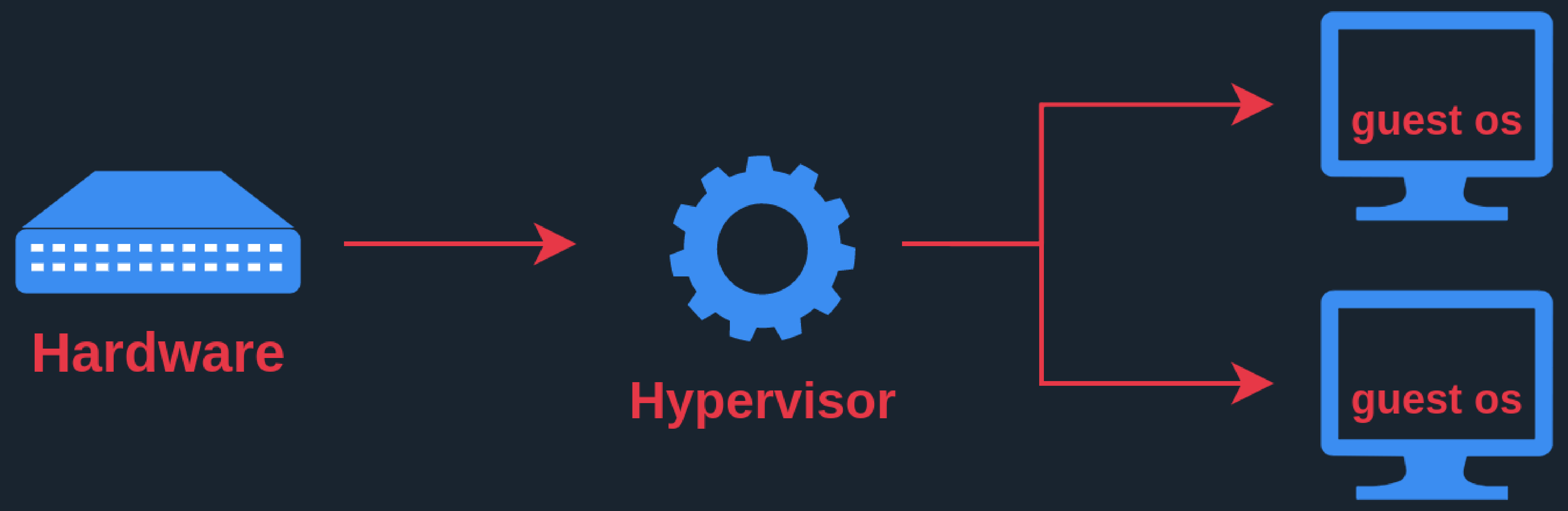
- “Bare metal” hypervisors (Type-1)
  - Runs directly on top of the hardware
  - Xen, Hyper-V, VMware ESXi



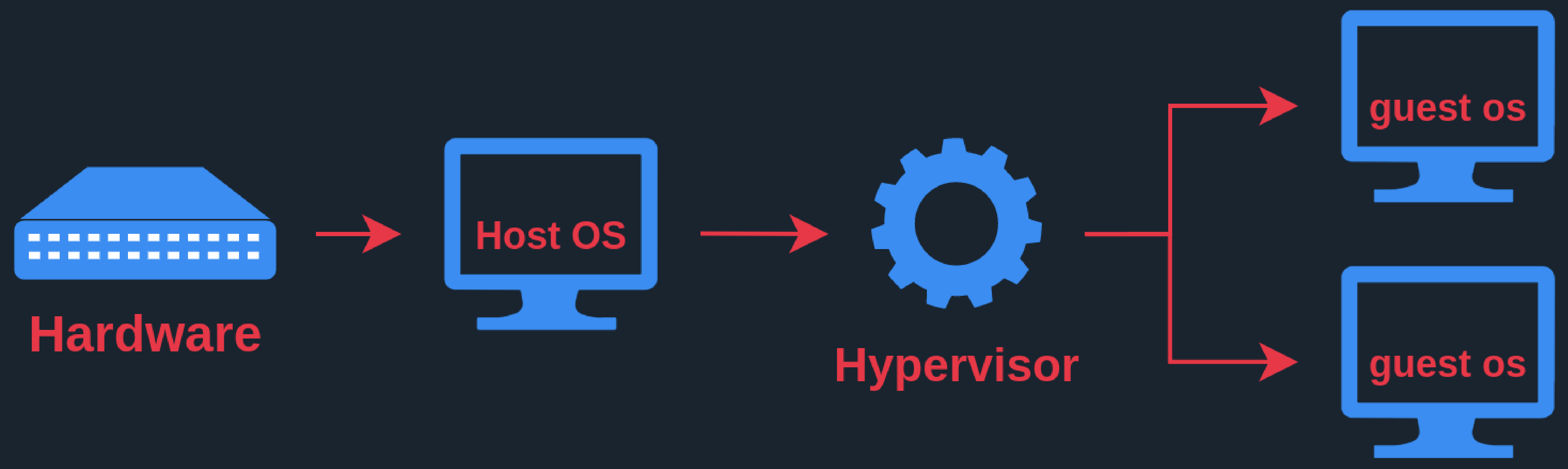
Software architecture

Type of hypervisors

- “Bare metal” hypervisors (Type-1)
  - Runs directly on top of the hardware
  - Xen, Hyper-V, VMware ESXi



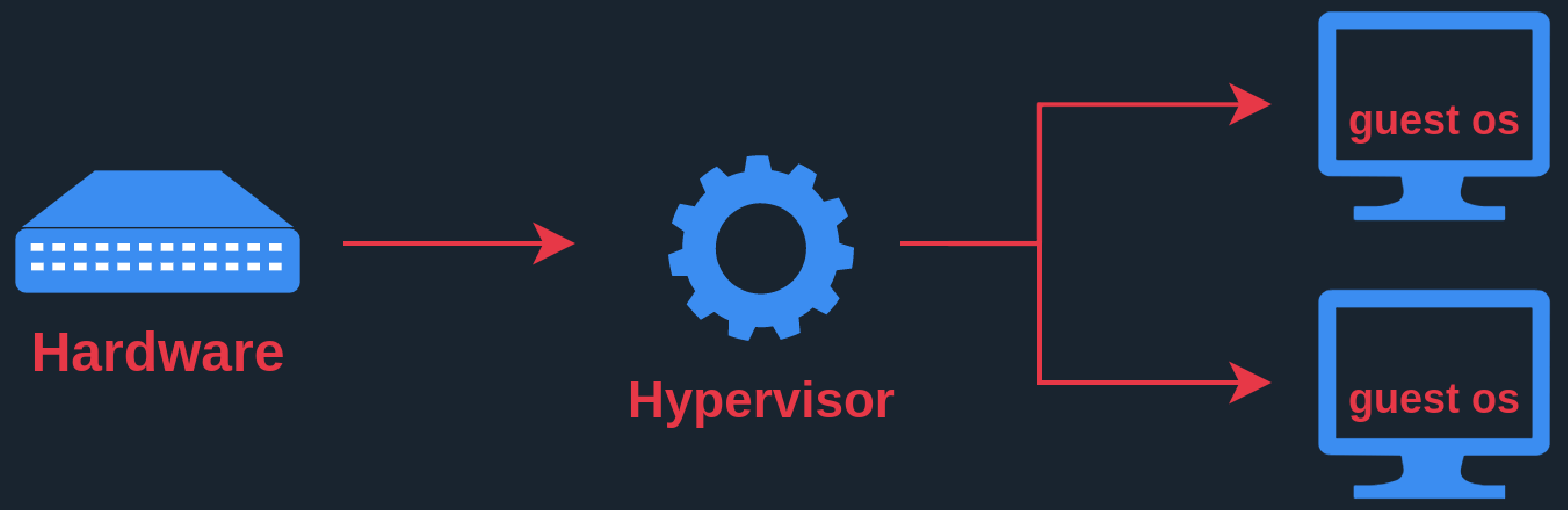
- “Hosted hypervisors” (Type-2)
  - Software running in an operating system
  - VMware Workstation, VirtualBox



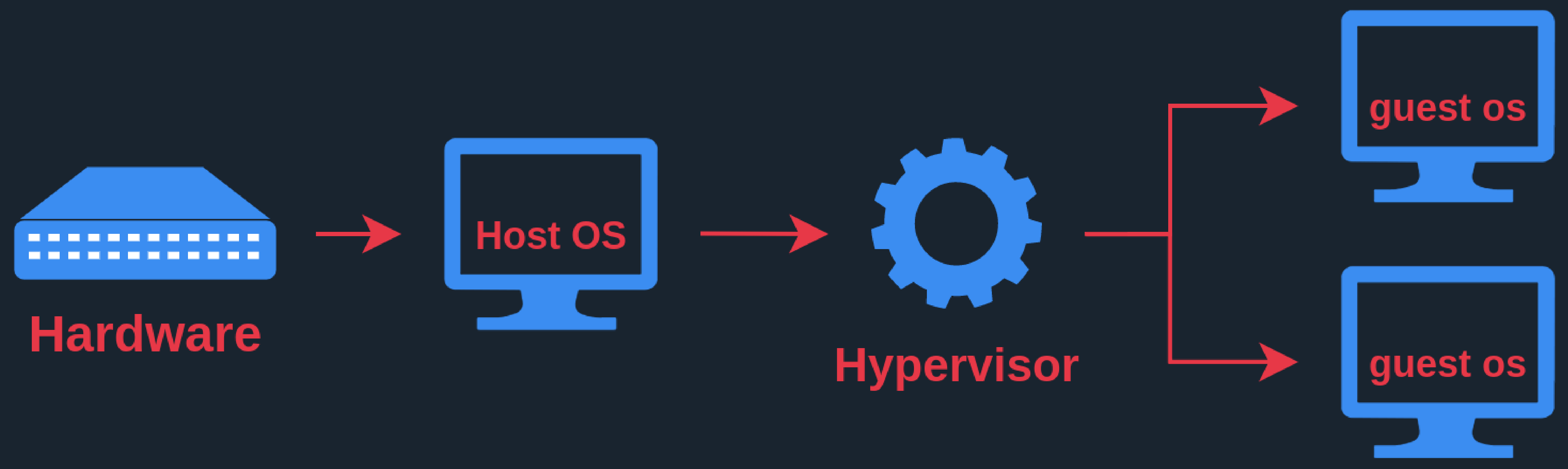
Software architecture

Type of hypervisors

- “Bare metal” hypervisors (Type-1)
  - Runs directly on top of the hardware
  - Xen, Hyper-V, VMware ESXi



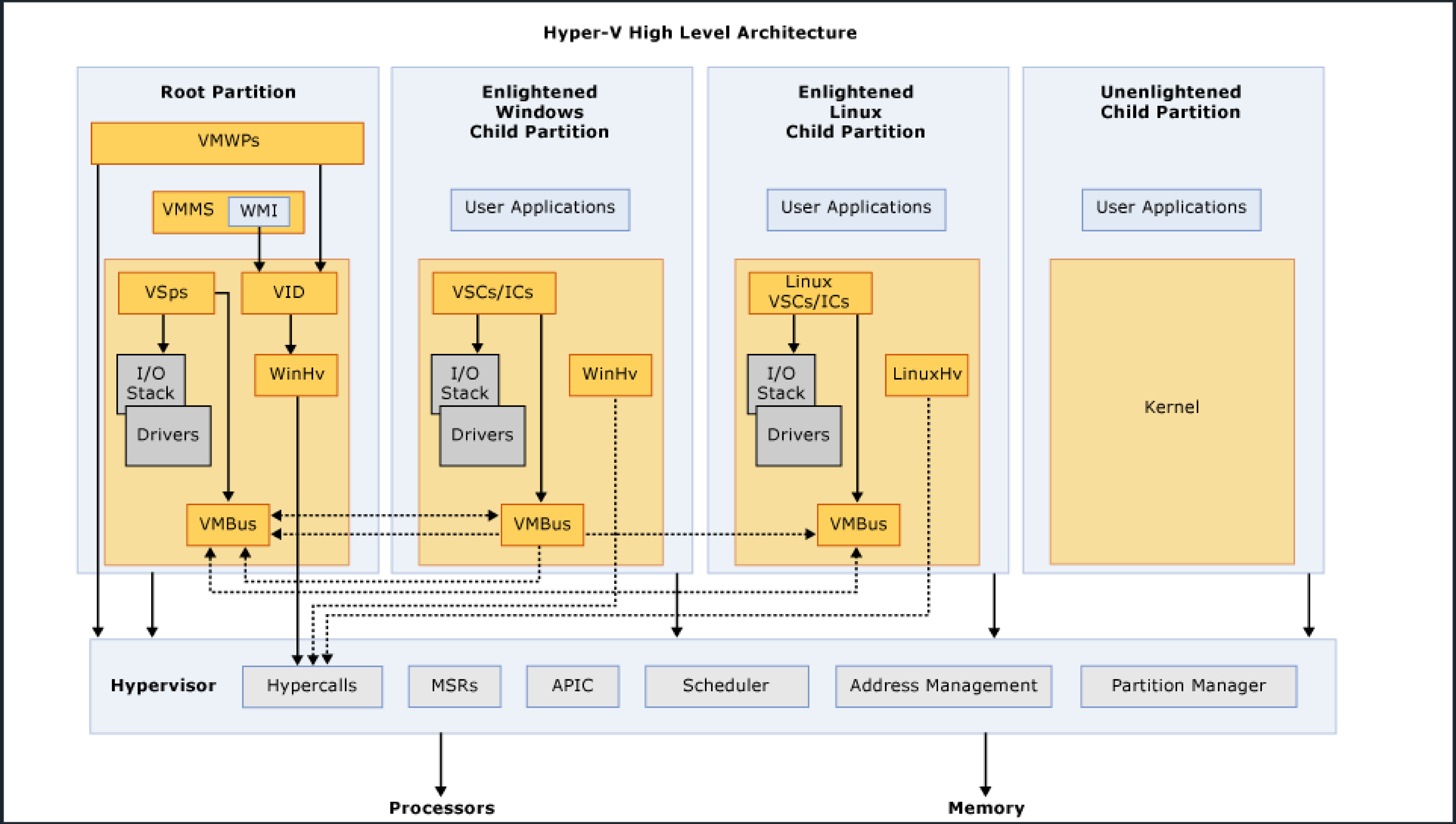
- “Hosted hypervisors” (Type-2)
  - Software running in an operating system
  - VMware Workstation, VirtualBox



- Not that many differences in the end…
  - Bare metal hypervisors have a base OS to handle applications
    - Windows for Hyper-V
    - Linux-like for VMware ESXi

Software architecture

Hyper-v

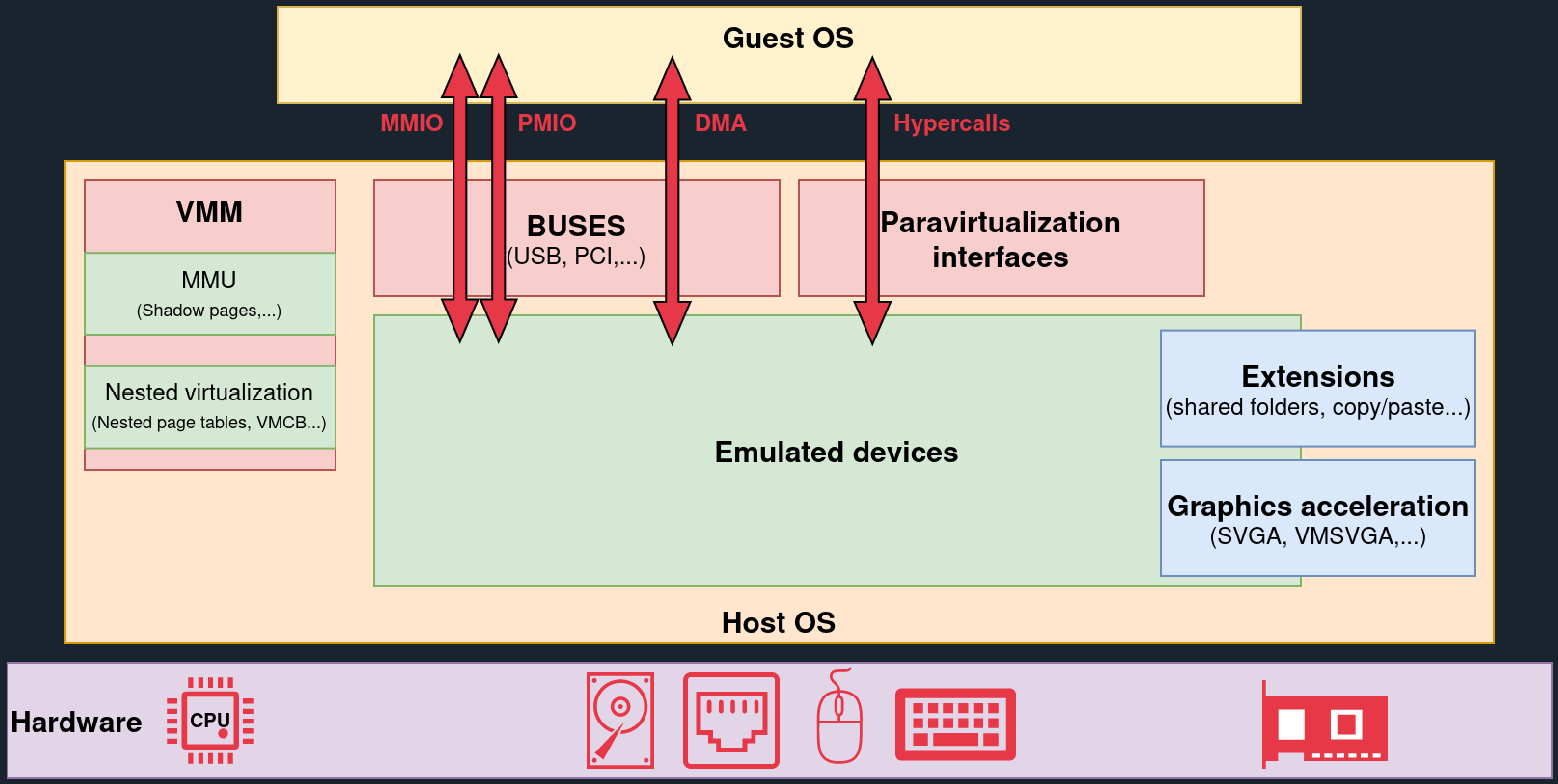


Virtualization components

Summary of the attack surface

Virtualization components

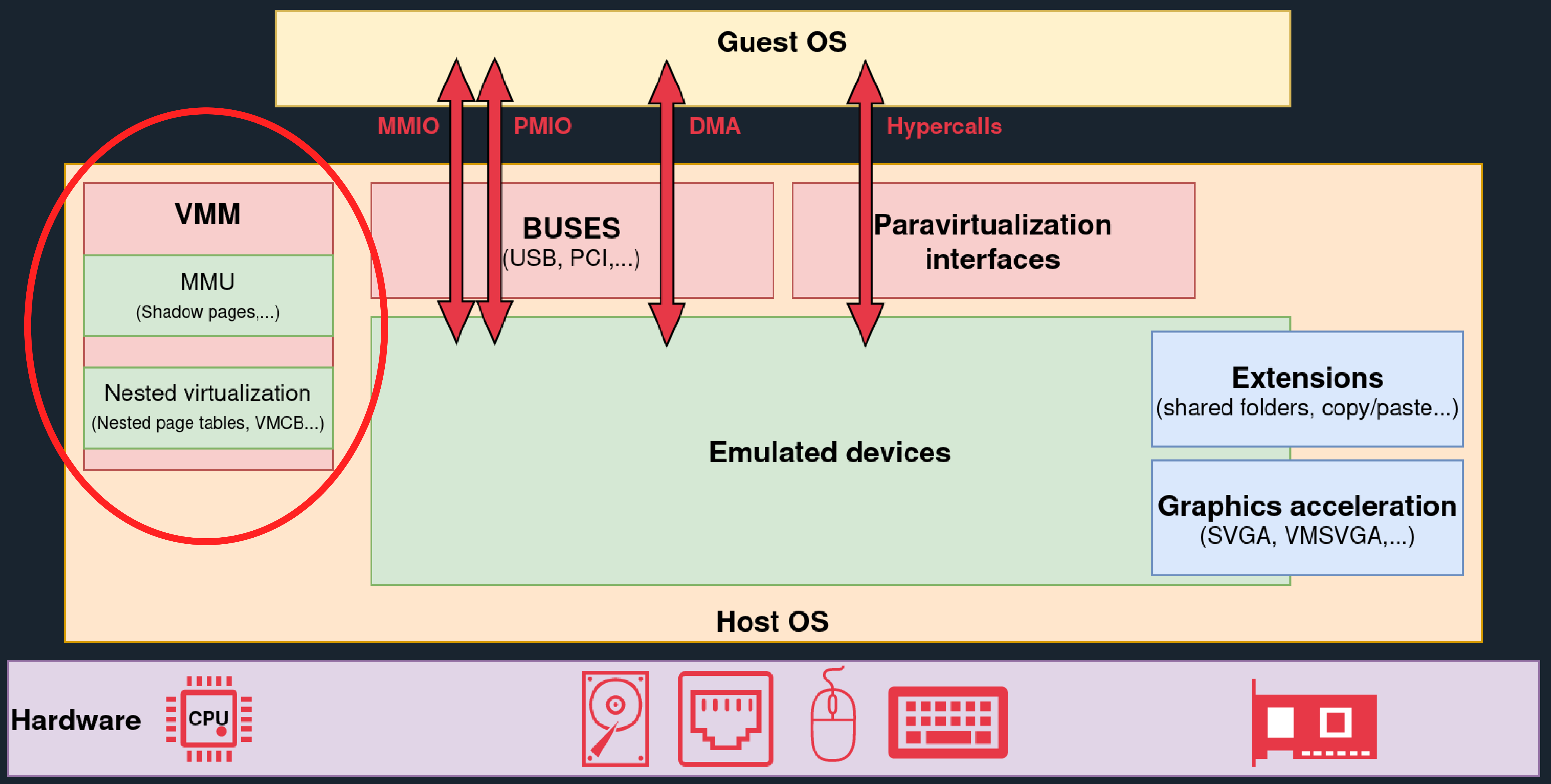
Attack surface overview



MMU

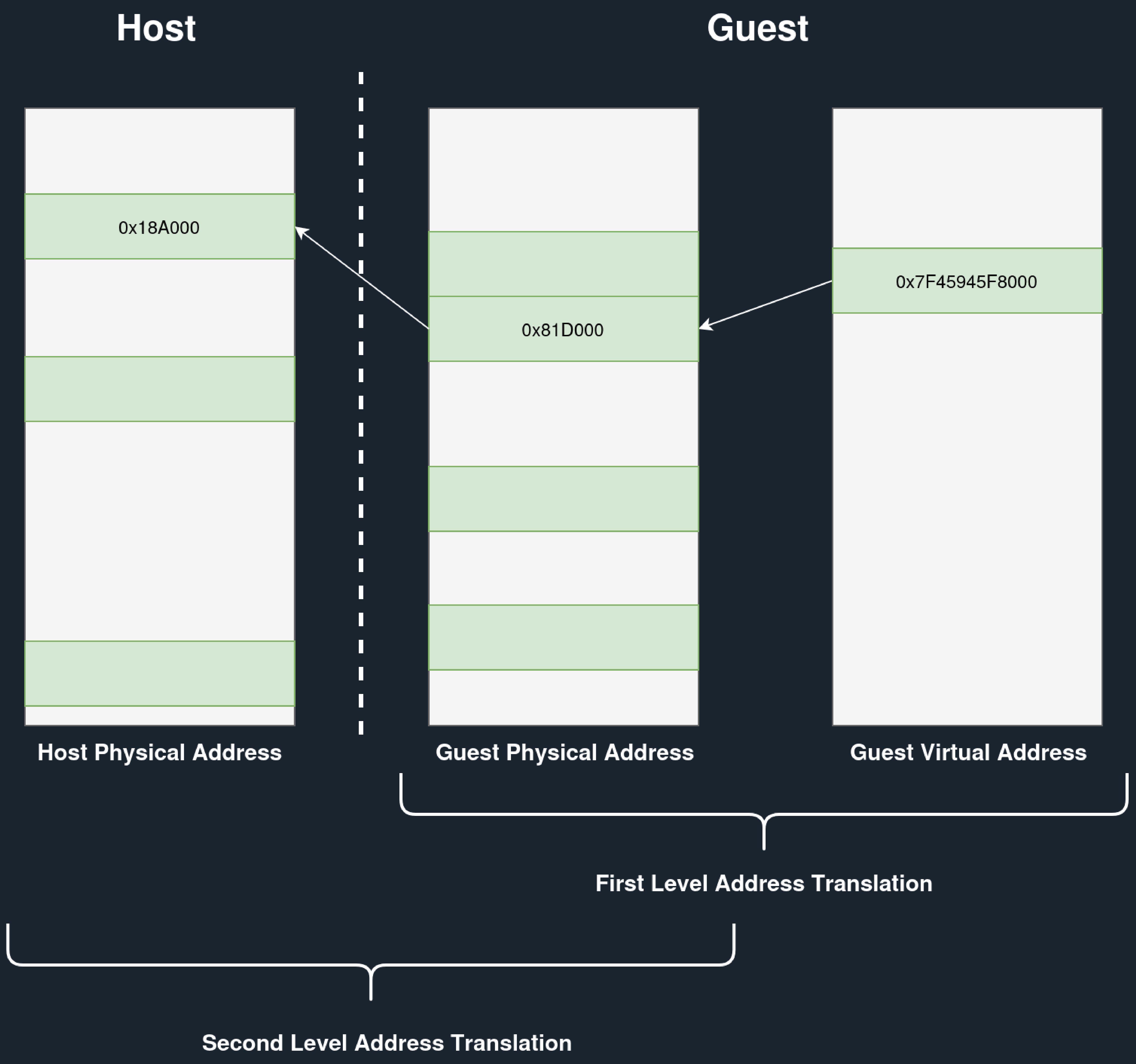
VMs memory management

- Hypervisors need to manage the guest physical memory
- Shadow Pages Tables
  - Mapping between **G**uest **V**irtual **A**ddresses and **H**ost **P**hysical **A**ddresses
  - Hardware acceleration brings **S**econd **L**evel **A**ddress **T**ranslation
    - Intel's **E**xtended **P**age **T**ables / AMD's **N**ested **P**age **T**ables



MMU

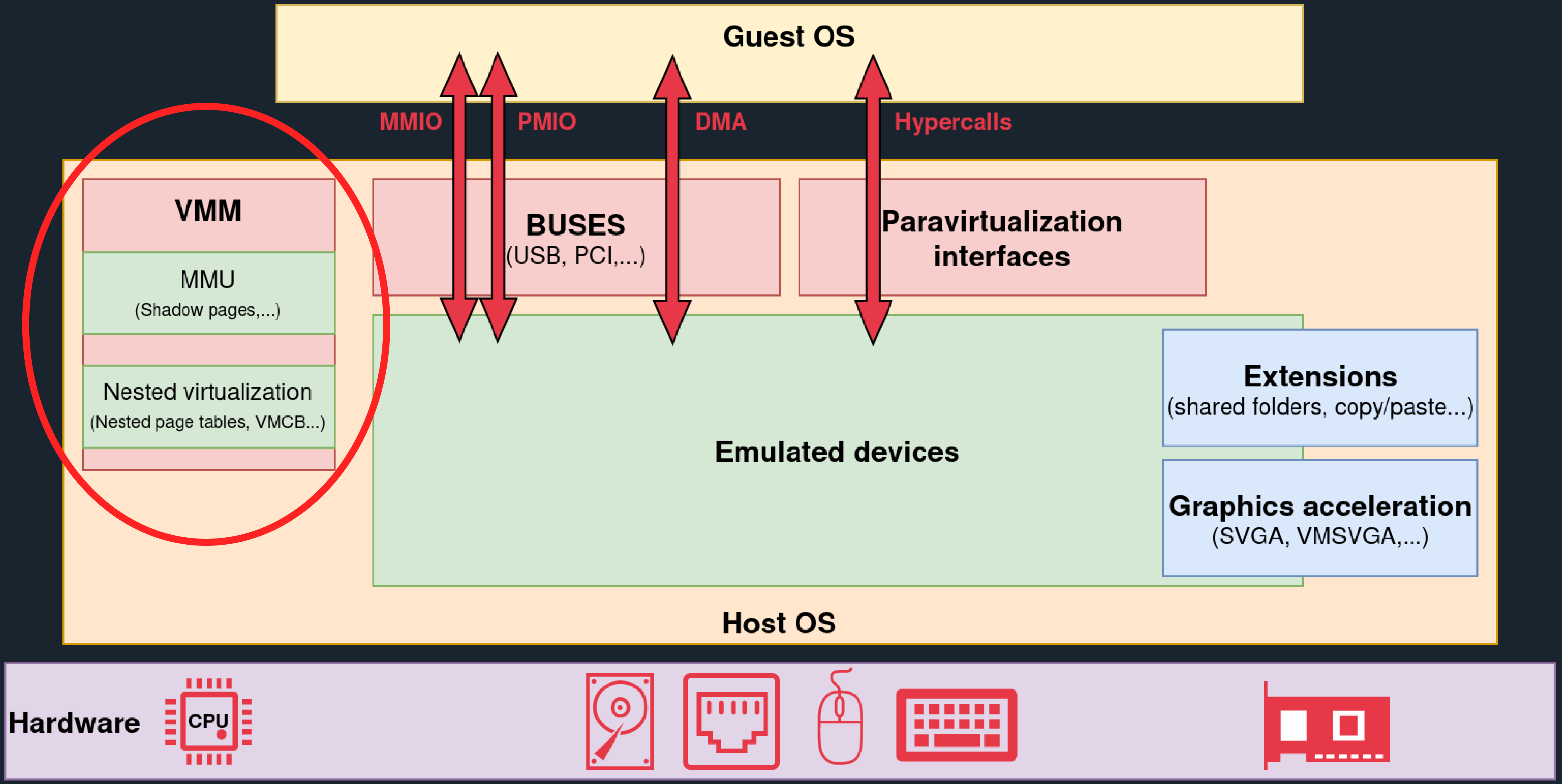
Second Level Address Translation (SLAT)



Nested Virtualization

Running a VM in a VM

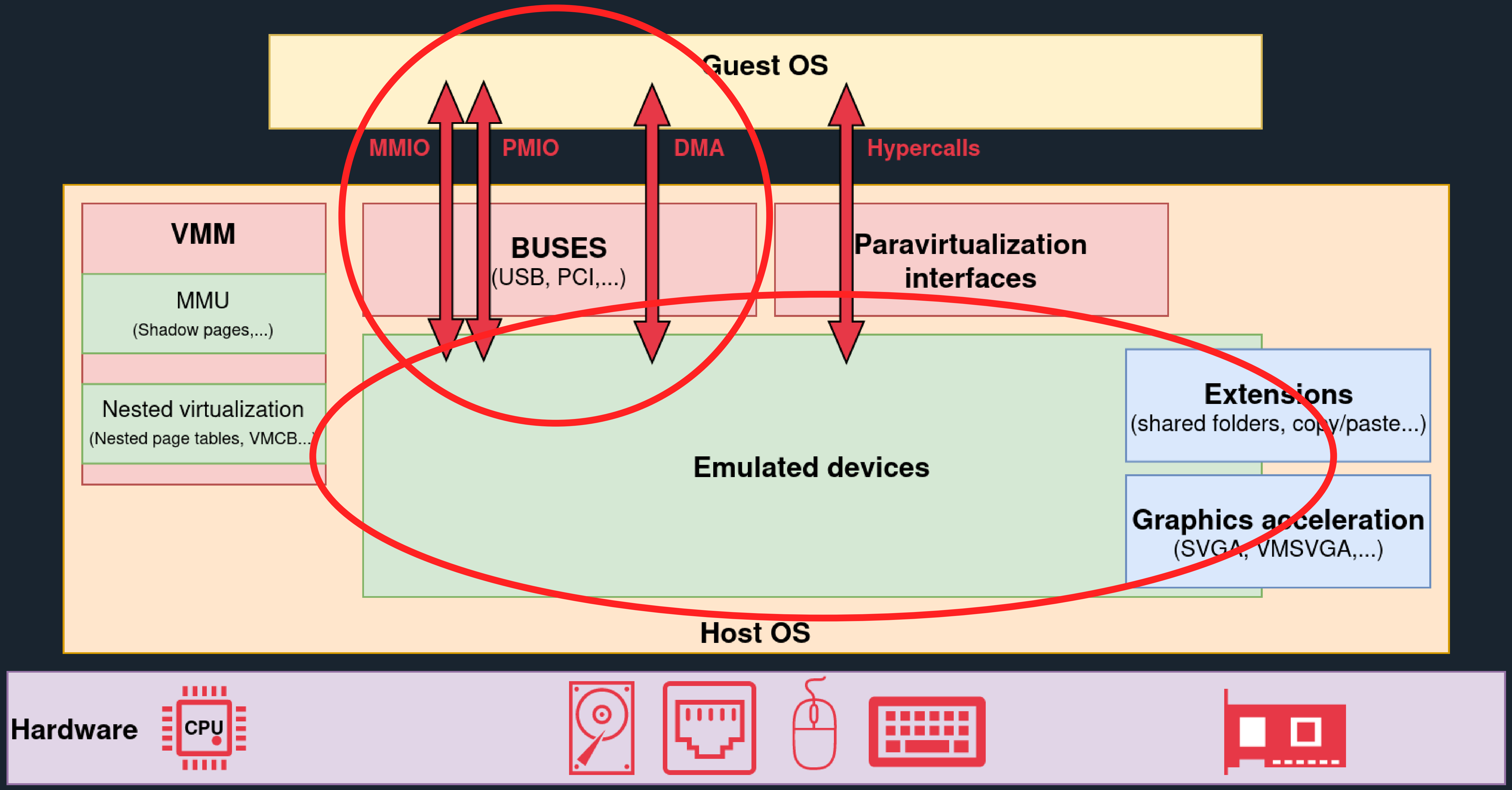
- Intel's VMCS / AMD's VMCB are the main data structures used by the hypervisors
- Hardware features need to be emulated
  - VMX instructions emulation
  - SLAT
  - APIC virtualization
  - VMCS shadowing
  - Hypervisors all handle these differently
- Complexity increases → attack surface increases



Buses

USB, PCI ...

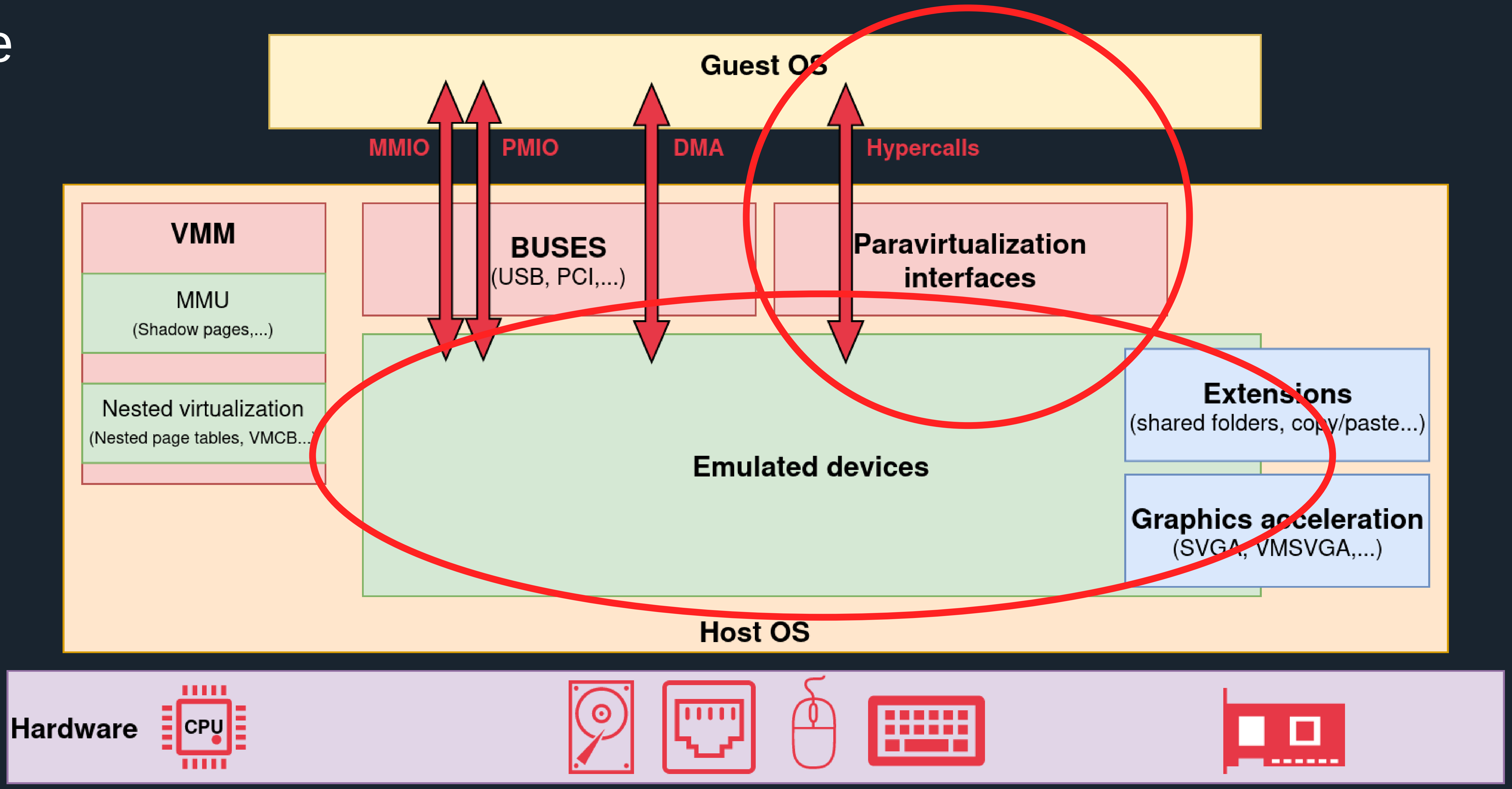
- A lot of hardware can be exposed via PCI or USB interfaces
- Either emulated devices or passthrough to hardware
- Very wide attack surface
  - Protocols, emulated devices
  - Very dependent of the configuration



Paravirtualized devices

Network cards, printers, disks...

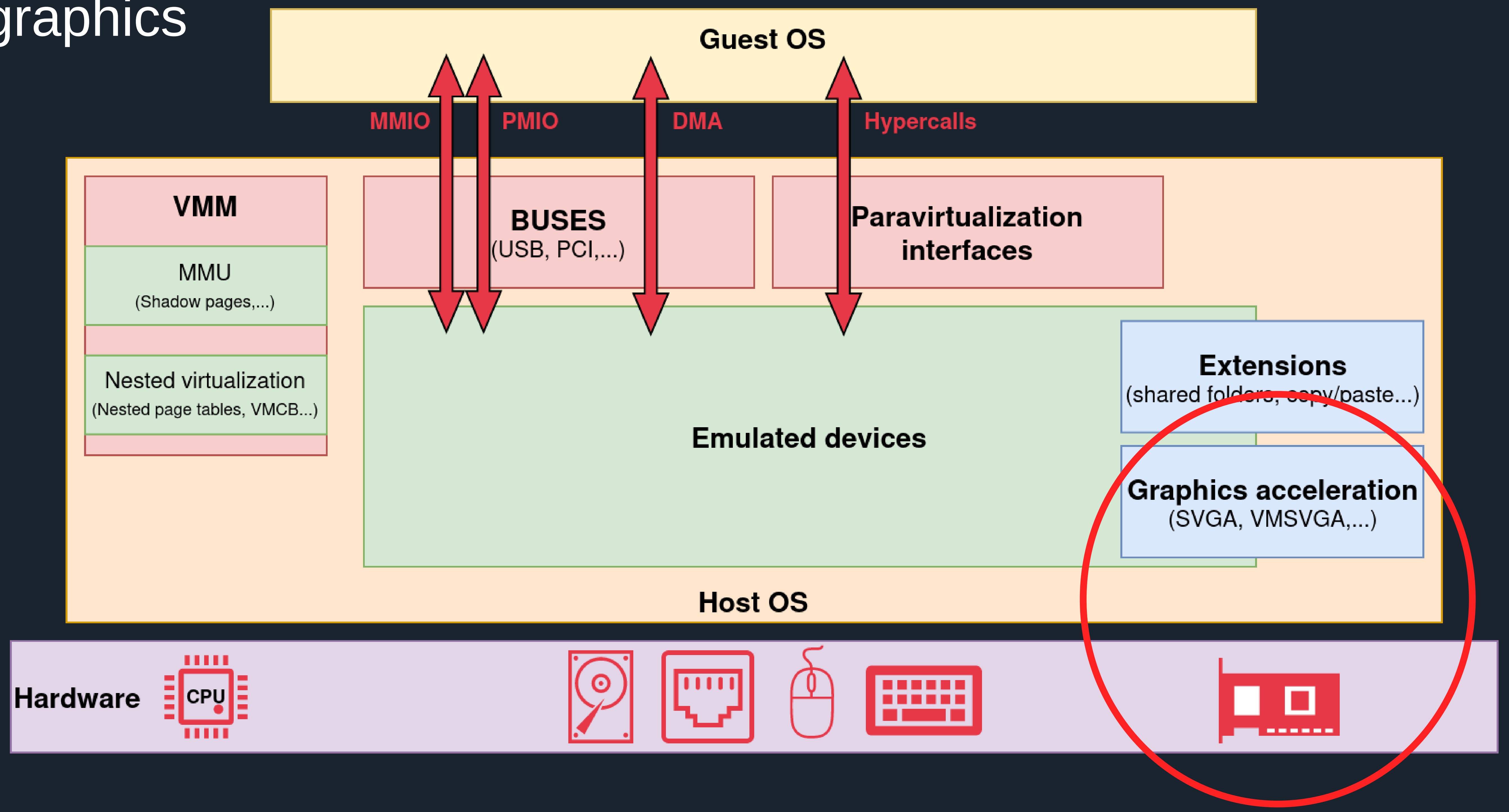
- Specific interfaces in the guest to communicate with the hypervisor
- The hypervisor has a lot of specific code to handle
- The OS embeds drivers for those devices
- Example devices that are often para-virtualized:
  - Network cards, Disks, Audio queues...



Graphics acceleration

SVGA, VMSVGA...

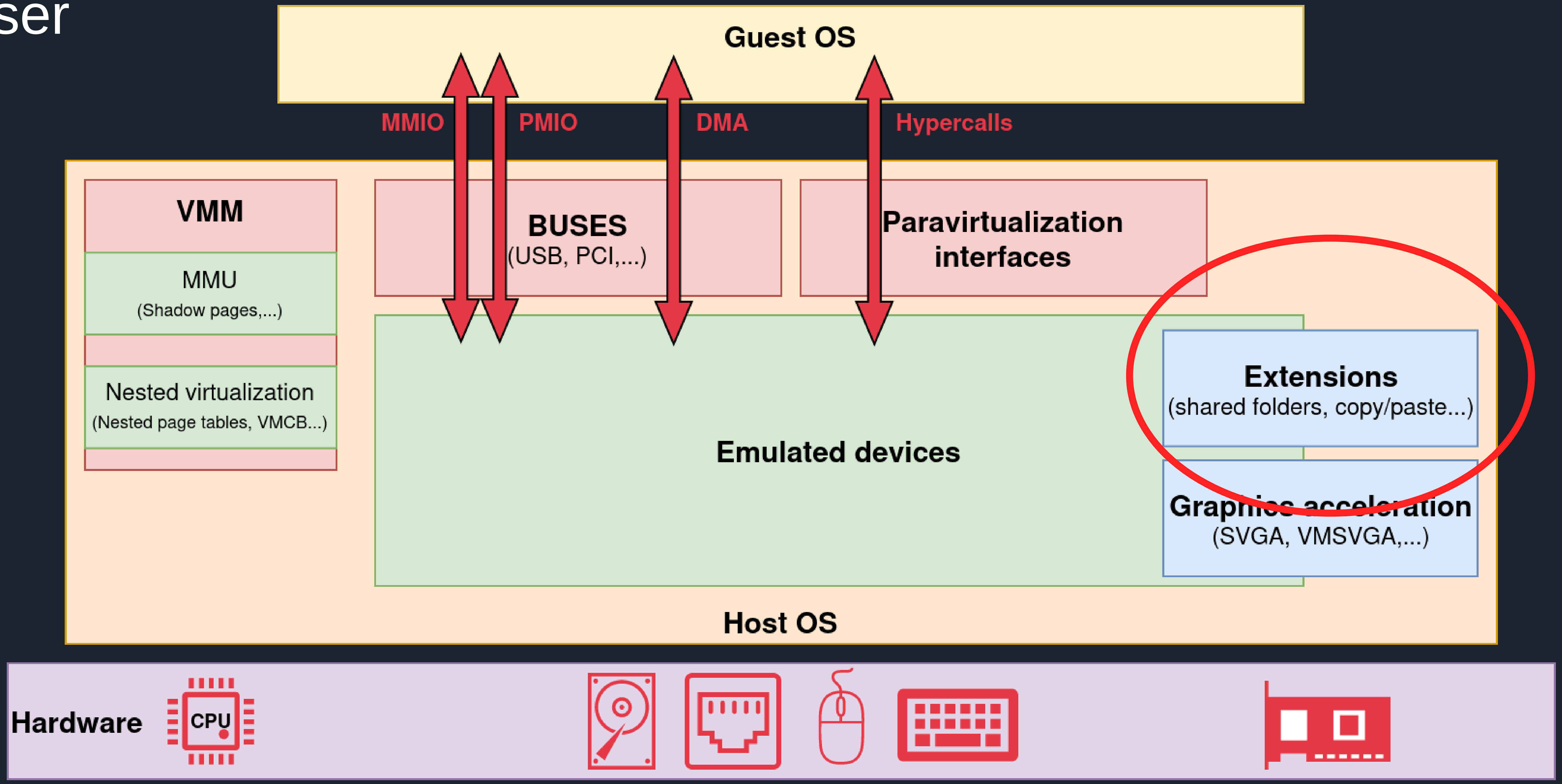
- Most hypervisors provide a way to accelerate graphics
- Allows to share the computing power of the GPU between multiple guests
- Complex interfaces to handle 3D graphics acceleration
- Examples:
  - SVGA on VMWare
  - VMSVGA on VirtualBox



Guest additions

Virtualization tools

- Modern hypervisors provide tools to help the user interact with the virtual machine
  - Copy/Paste
  - Drag and Drop
  - Shared Folders
- Mostly installed on hosted hypervisors
  - VirtualBox, VMware Workstation
- Used to be a very popular way to attack the hypervisor
  - Not too complex, dangerous features, error prone



Conclusion on virtualization

Attack surface

- CPU and MMU components represent a complex attack surface
  - Both from the defender and attacker’s point-of-view
  - Vulnerabilities in those components are very powerful
    - Break the MMU or CPU isolation and you control the host
    - But very complex and time-consuming
  - It does NOT represents the main attack surface chosen by attackers

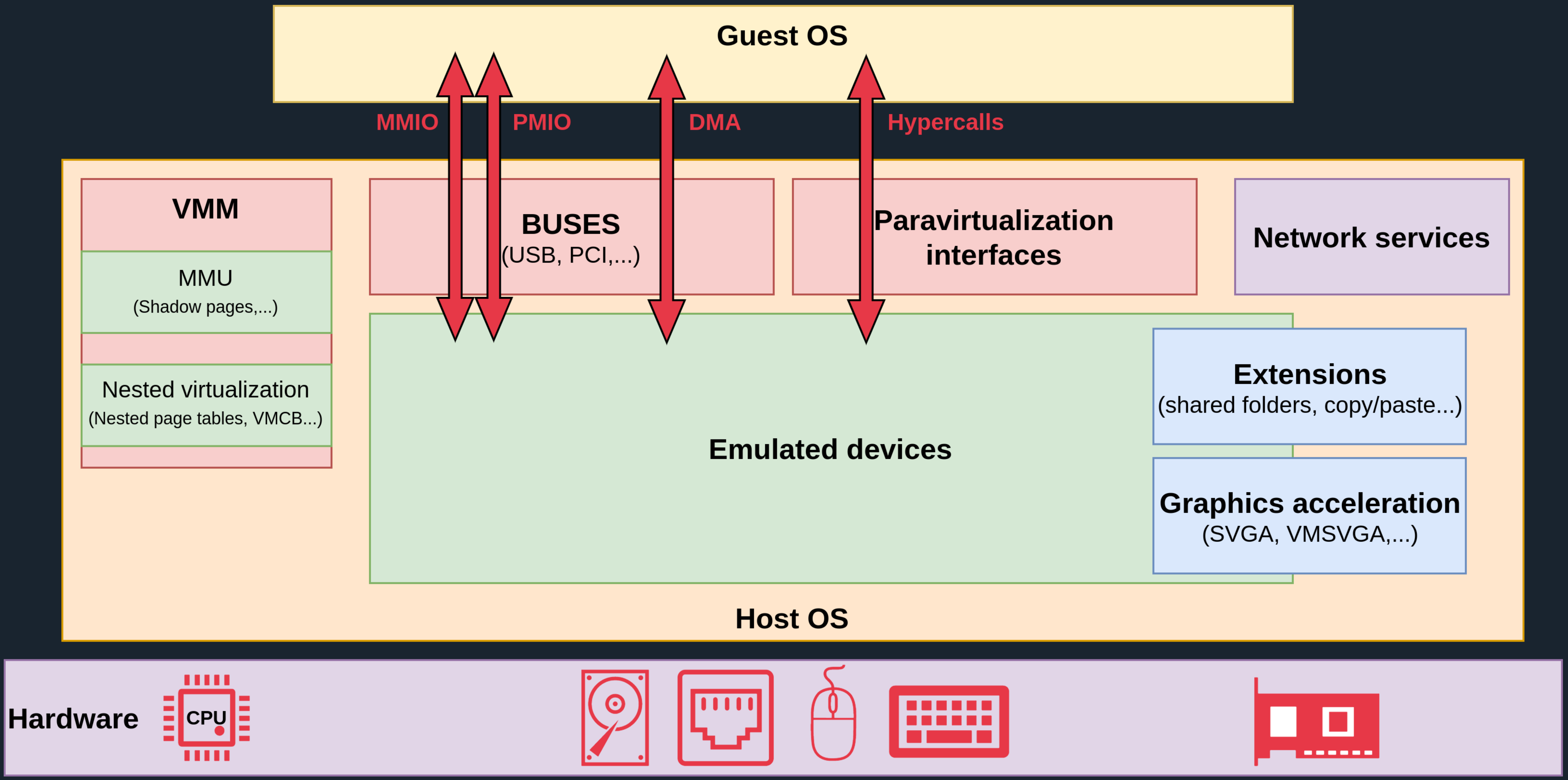
- Emulated and paravirtualized devices still represent the main attack surface
  - A lot of code, less complex
    - “Classic C bugs”: Buffer overflow, integer overflow, use-after-free…
  - You don’t have to fully understand the complex virtualization mechanisms to find and exploit bugs
  - But less style points scored when disclosing a bug !

Down the rabbit hole

History of virtualization bugs

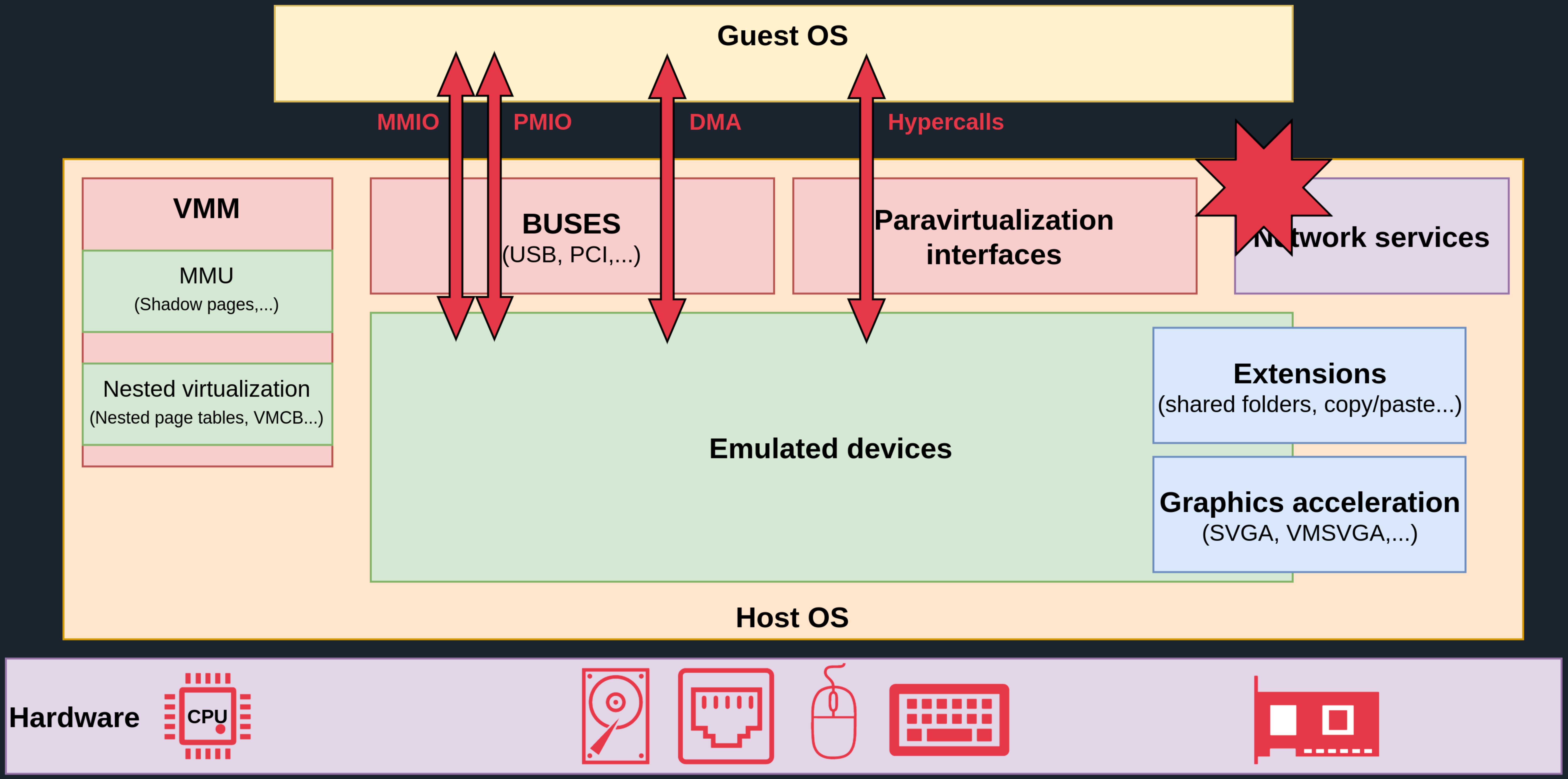
Attack surface

Hypervisor's attack surface



Network service bug

Overview



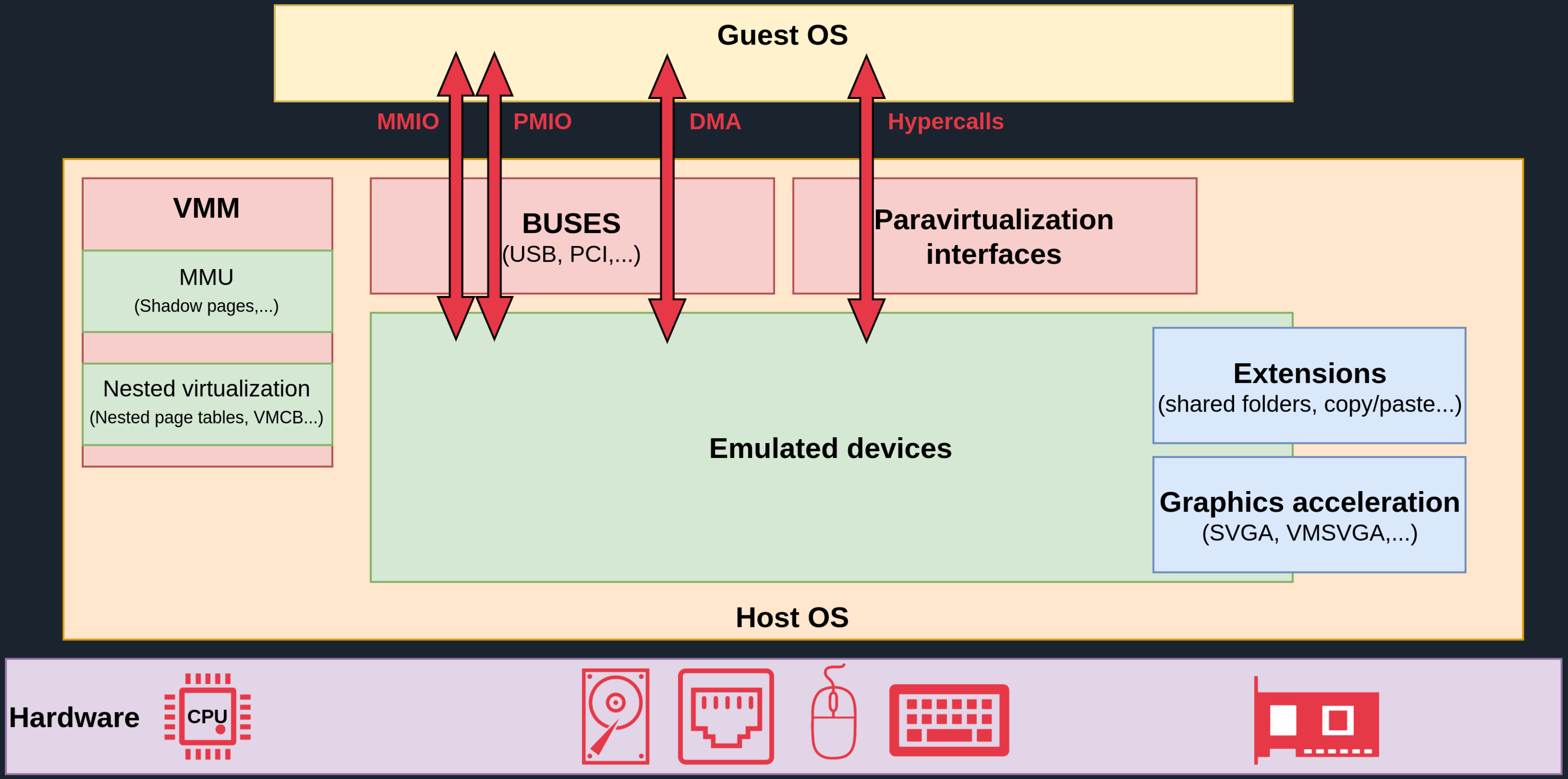
Network service bug

CVE-2019-5544: Remote Code Execution in VMware ESXi

- CVE-2019-5544: RCE in service OpenSLP of ESXi
  - Network service running on the host of ESXi
  - Open-source implementation of the **S**ervice **L**ocation **P**rotocol (**SLP**)
- Was reachable by default from the VM **and** on the administration interface
- Heap overflow exploited at the TianfuCup 2019
  - Multiple bugs in the same service were found after the competition
    - CVE-2020-3992: Use-After-Free
    - CVE-2021-21974: Heap overflow
- Was actively exploited in the wild as a 1-day
  - ESXiArgs: ransomware on ESXi
  - Mostly exploited on the administration interface and not as VME
- Not a **virtualization** bug
  - It's not the kind of bugs we are interested in !

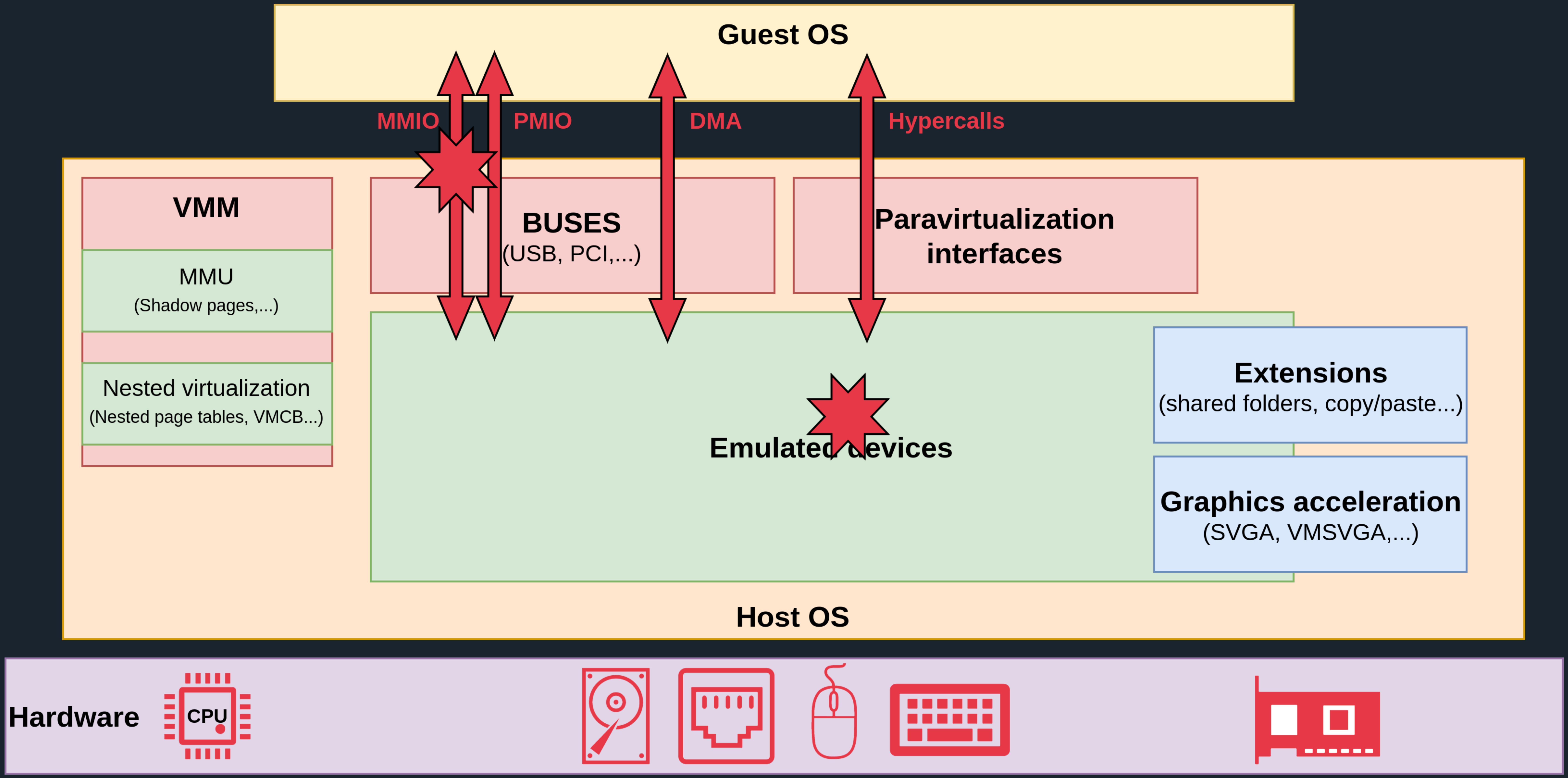
Attack surface

Hypervisor's attack surface



Attack surface

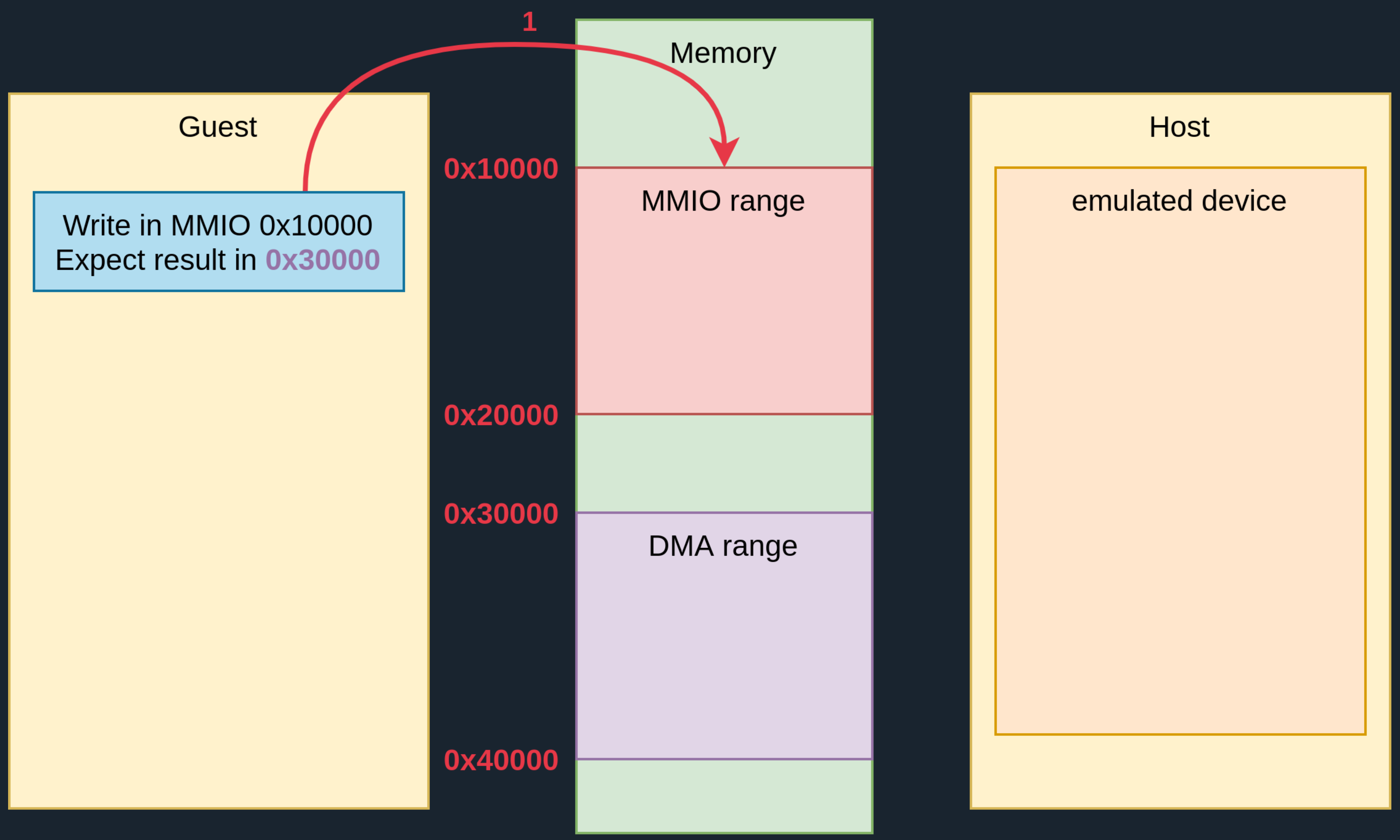
Hypervisor’s attack surface



MMIO

Expected behavior

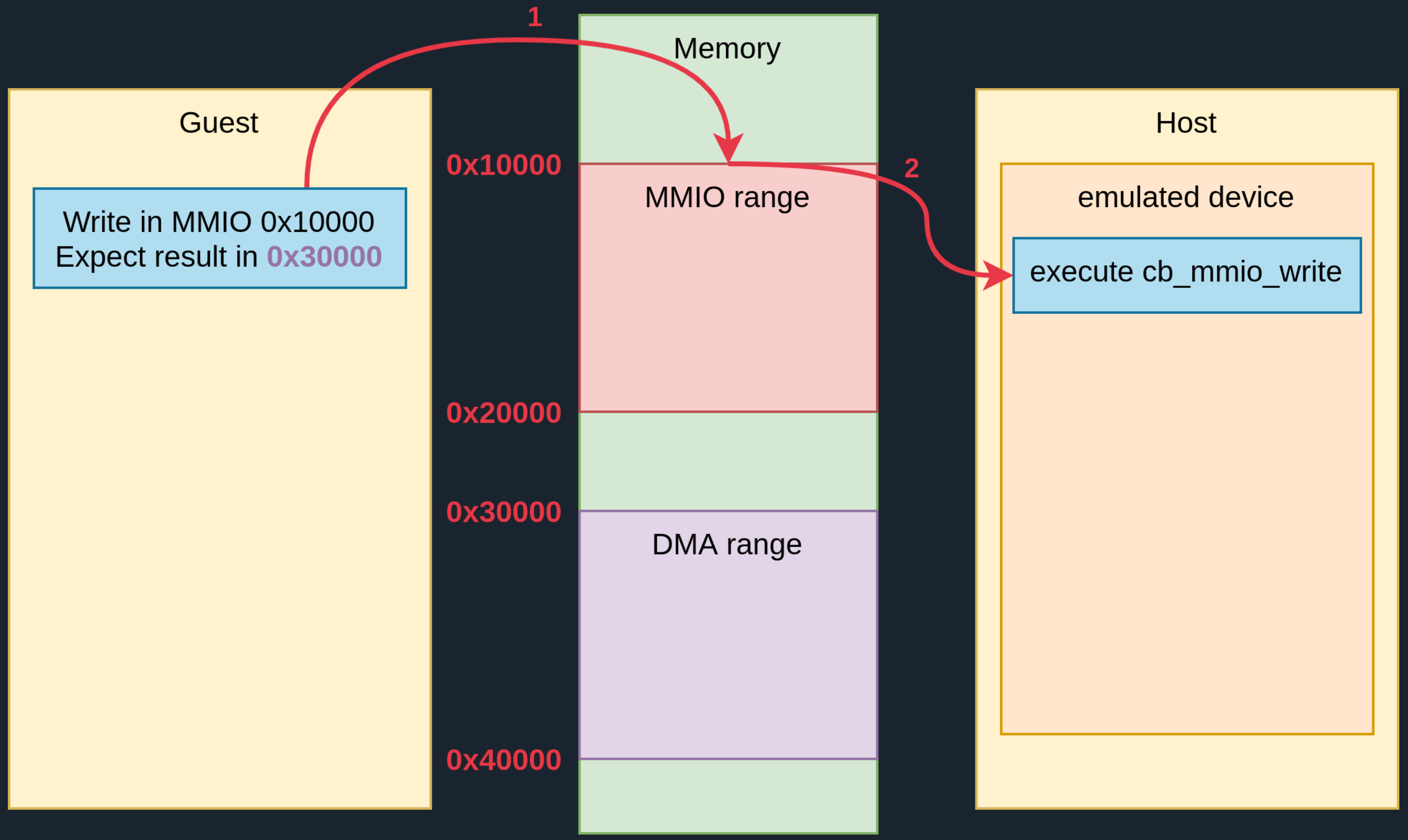
- Guest can trigger callbacks by writing in MMIO
  - “Memory Mapped IO”
  - Sometimes, guest can expect data to be written through DMA
  - Usually provides the DMA’s buffer address



MMIO

Expected behavior

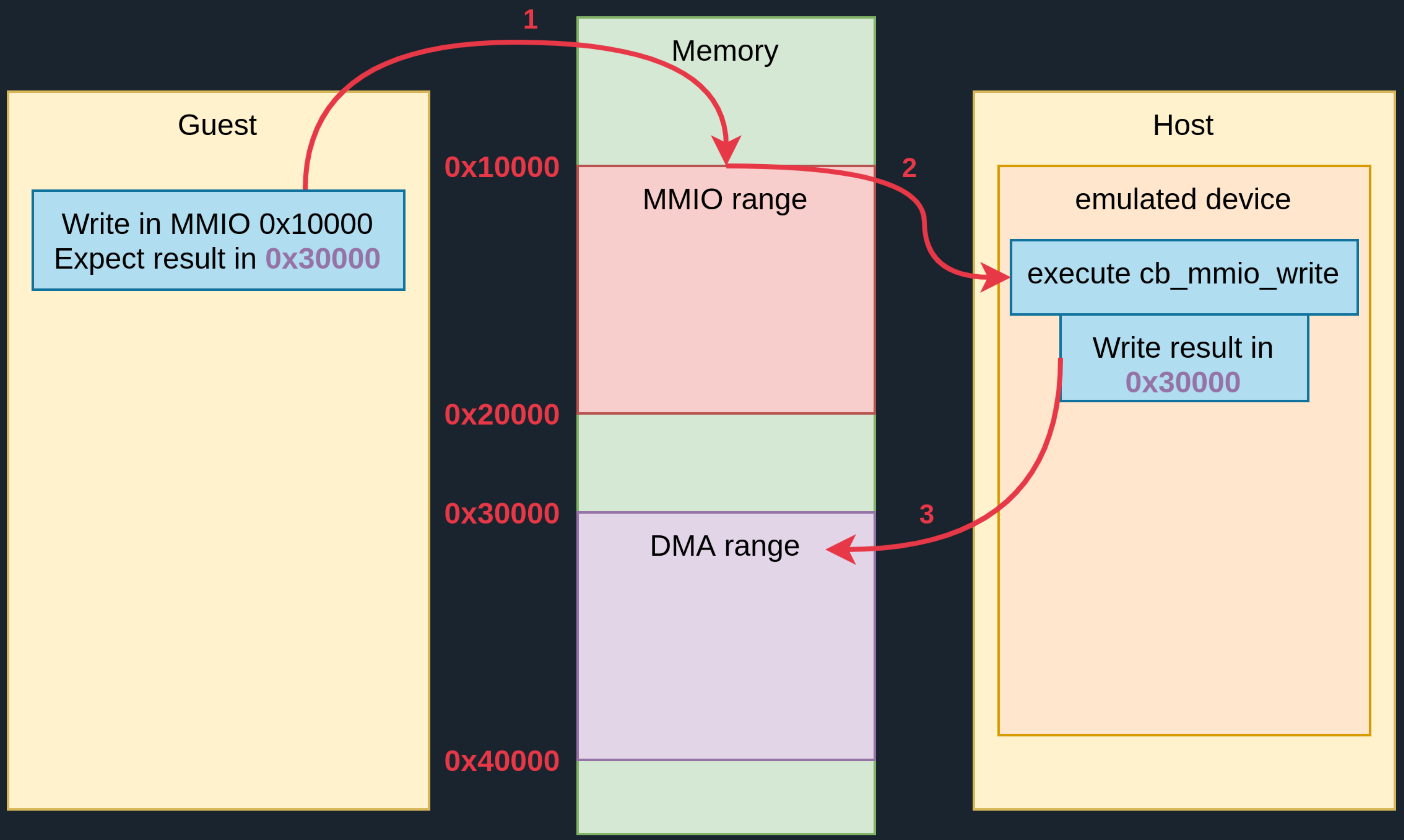
- Guest can trigger callbacks by writing in MMIO
  - “Memory Mapped IO”
  - Sometimes, guest can expect data to be written through DMA
  - Usually provides the DMA’s buffer address
- Host executes the callback based on which address was written



MMIO

Expected behavior

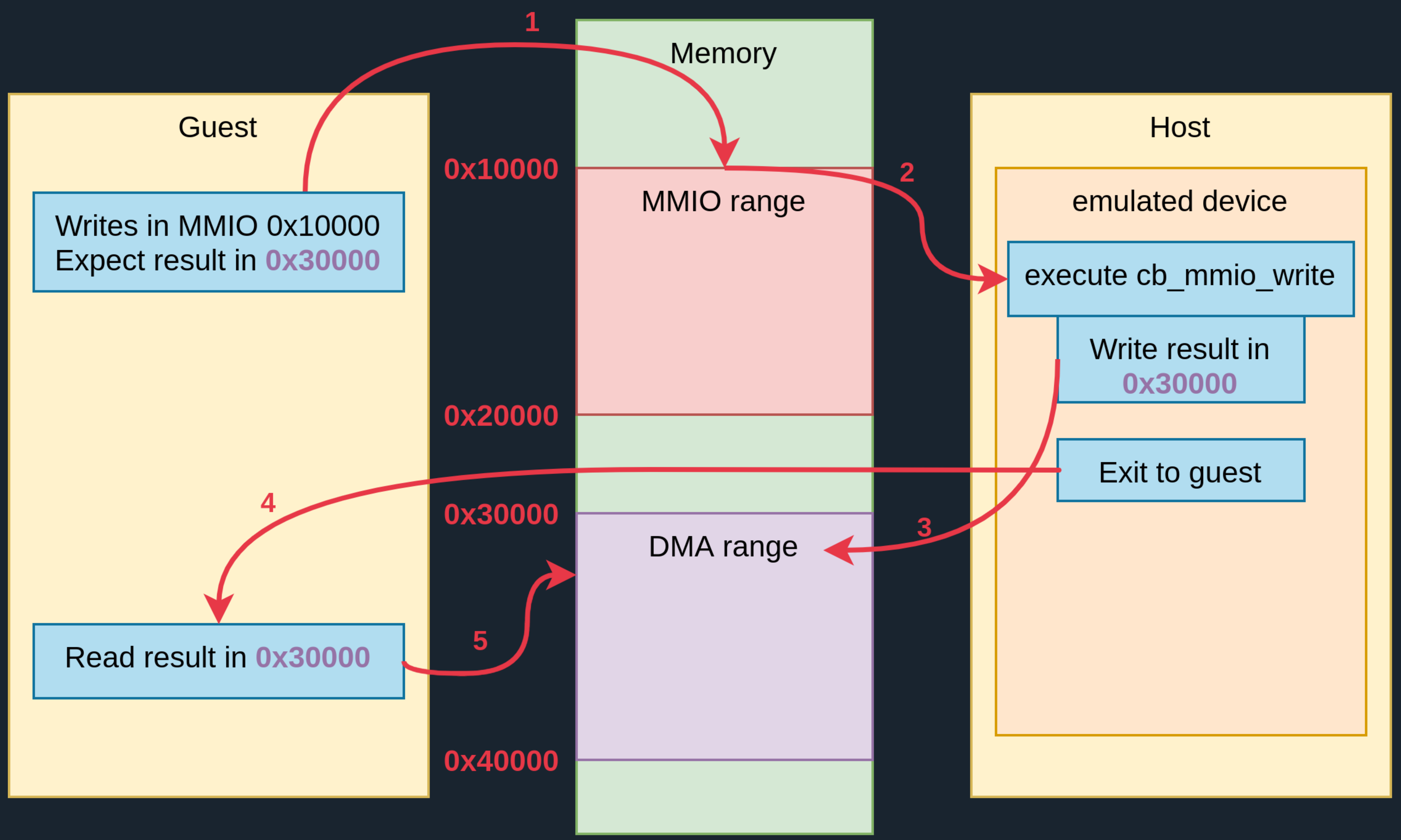
- Guest can trigger callbacks by writing in MMIO
  - “Memory Mapped IO”
  - Sometimes, guest can expect data to be written through DMA
  - Usually provides the DMA’s buffer address
- Host executes the callback based on which address was written
  - Host writes result in the provided DMA buffer



MMIO

Expected behavior

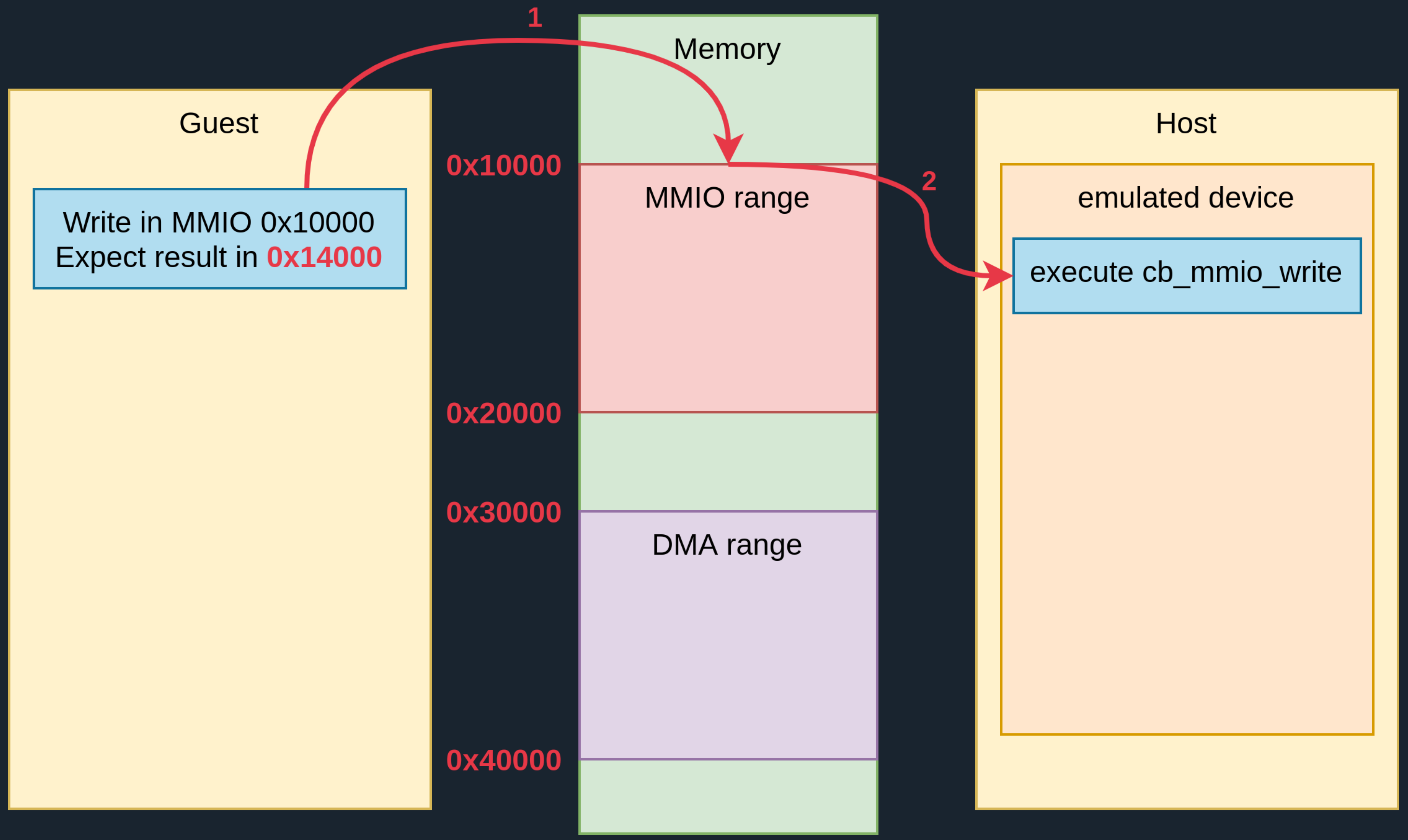
- Guest can trigger callbacks by writing in MMIO
  - “Memory Mapped IO”
  - Sometimes, guest can expect data to be written through DMA
  - Usually provides the DMA’s buffer address
- Host executes the callback based on which address was written
  - Host writes result in the provided DMA buffer
- Host gives back execution to guest
- Guest can read result from DMA



Recursive MMIO

A vulnerability pattern

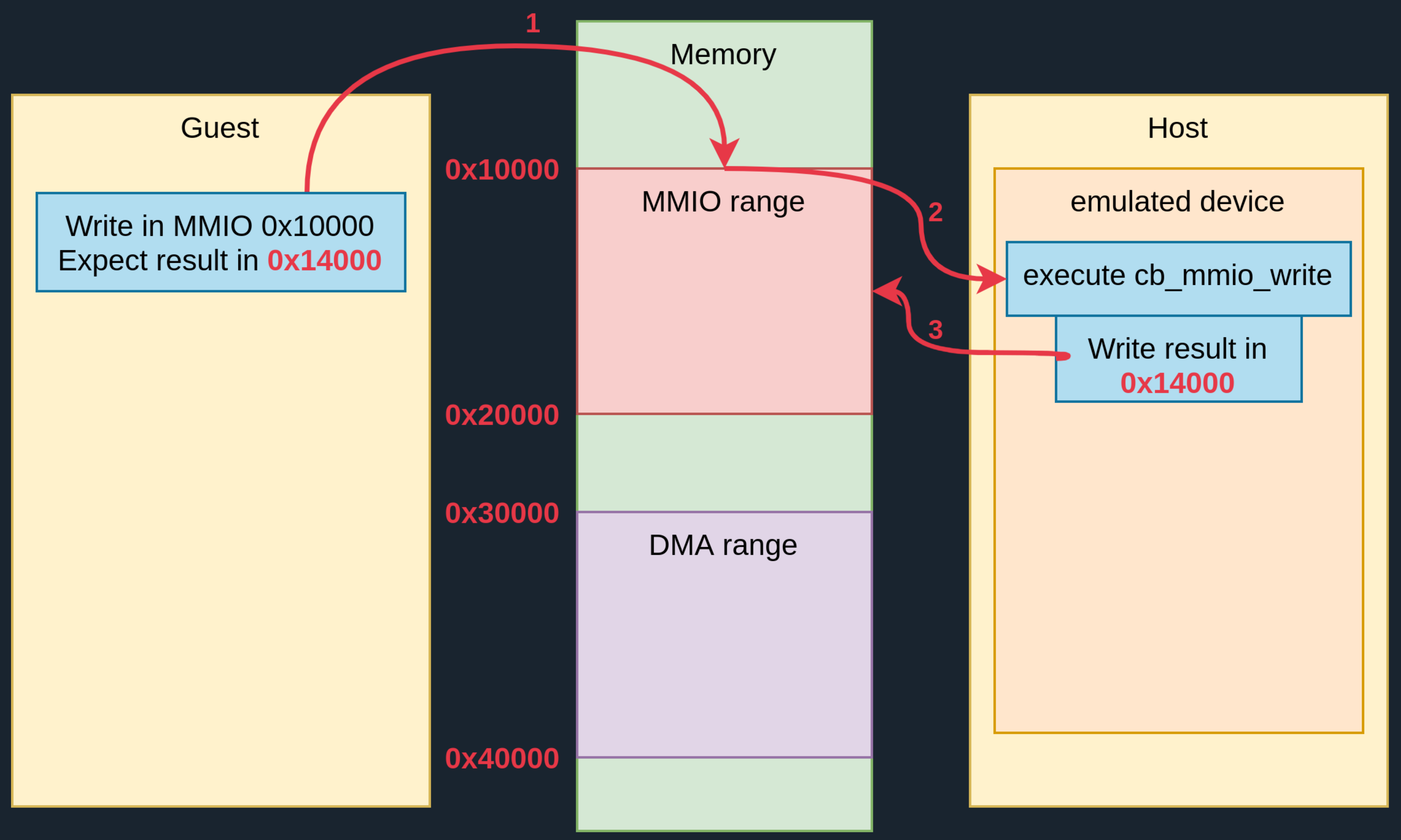
- What if the guest provides a MMIO address to the host ?



Recursive MMIO

A vulnerability pattern

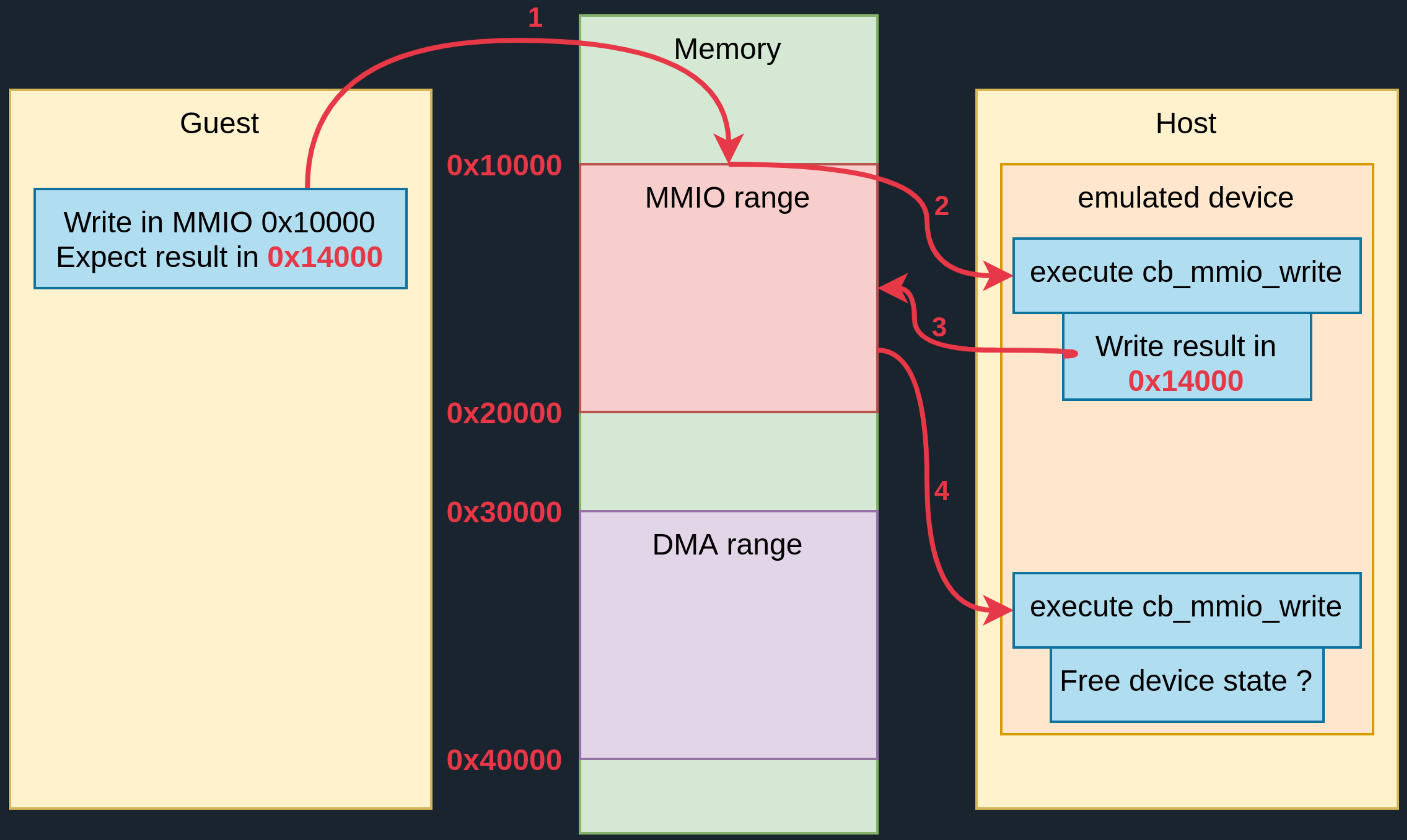
- What if the guest provides a MMIO address to the host ?
- Host will write back in the MMIO…



Recursive MMIO

A vulnerability pattern

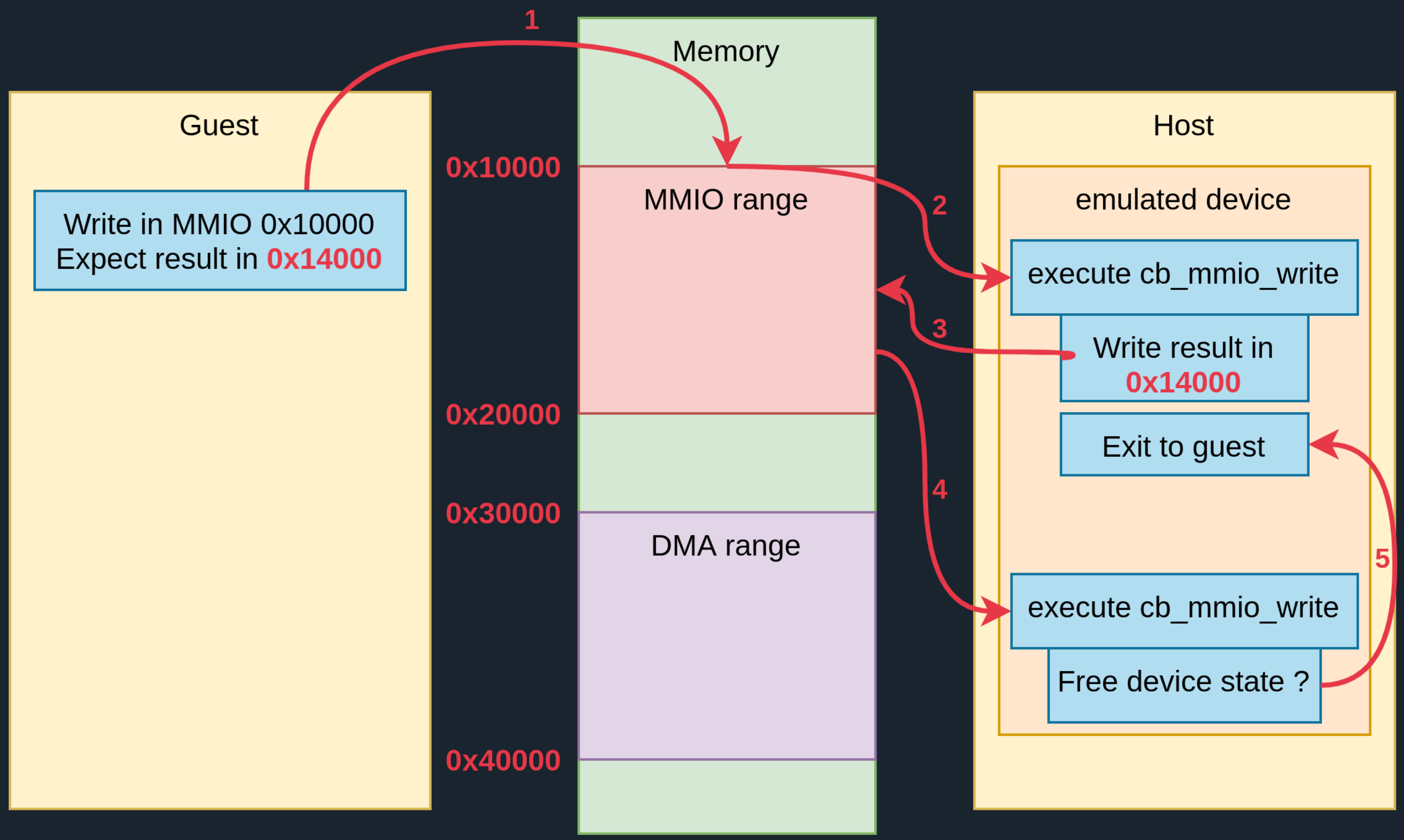
- What if the guest provides a MMIO address to the host ?
- Host will write back in the MMIO…
- Triggering another MMIO handler !
  - Depending on the callback, some device structures might be freed



Recursive MMIO

A vulnerability pattern

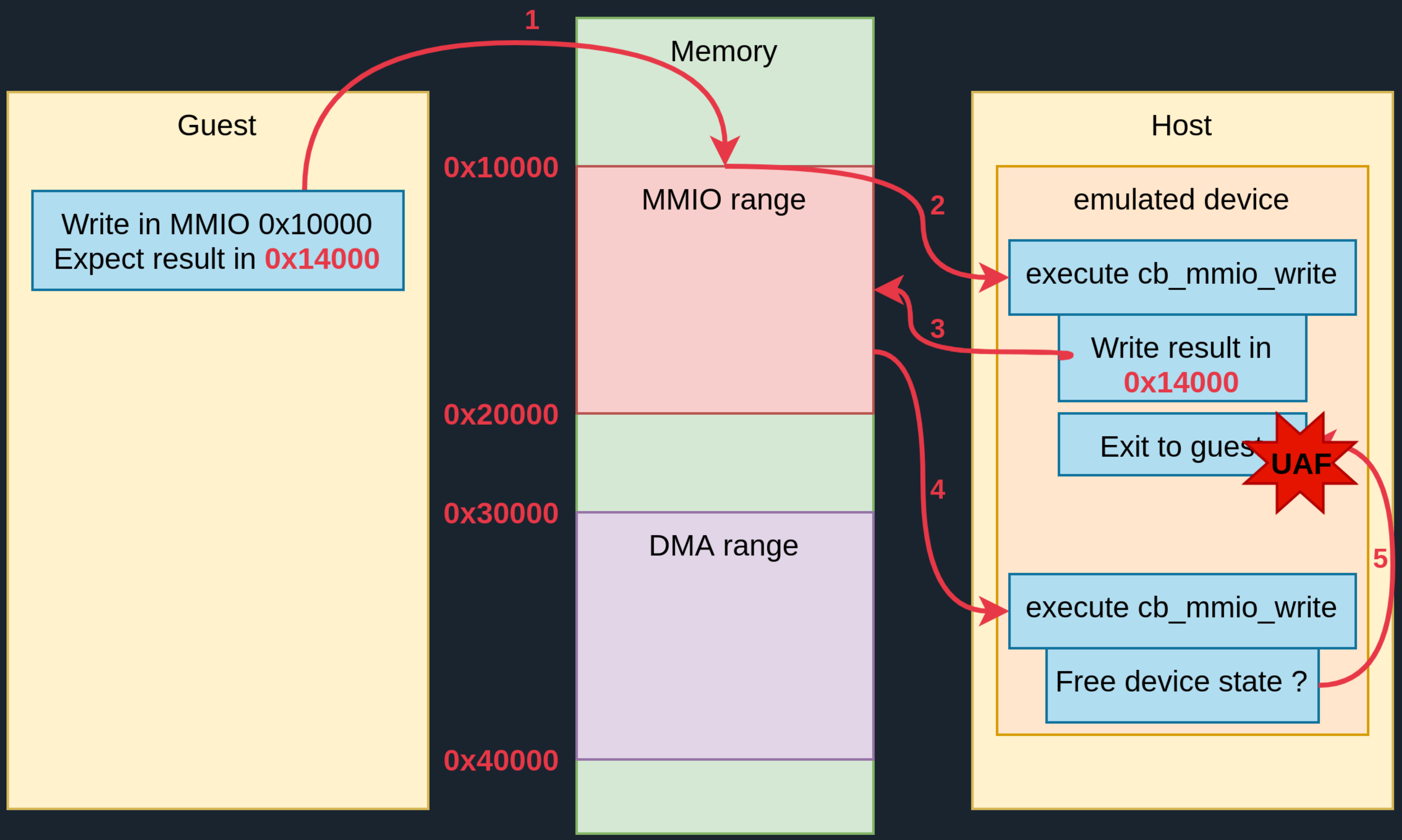
- What if the guest provides a MMIO address to the host ?
- Host will write back in the MMIO…
- Triggering another MMIO handler !
    - Depending on the callback, some device structures might be freed
- Execution is given back to the first MMIO handler…



Recursive MMIO

A vulnerability pattern

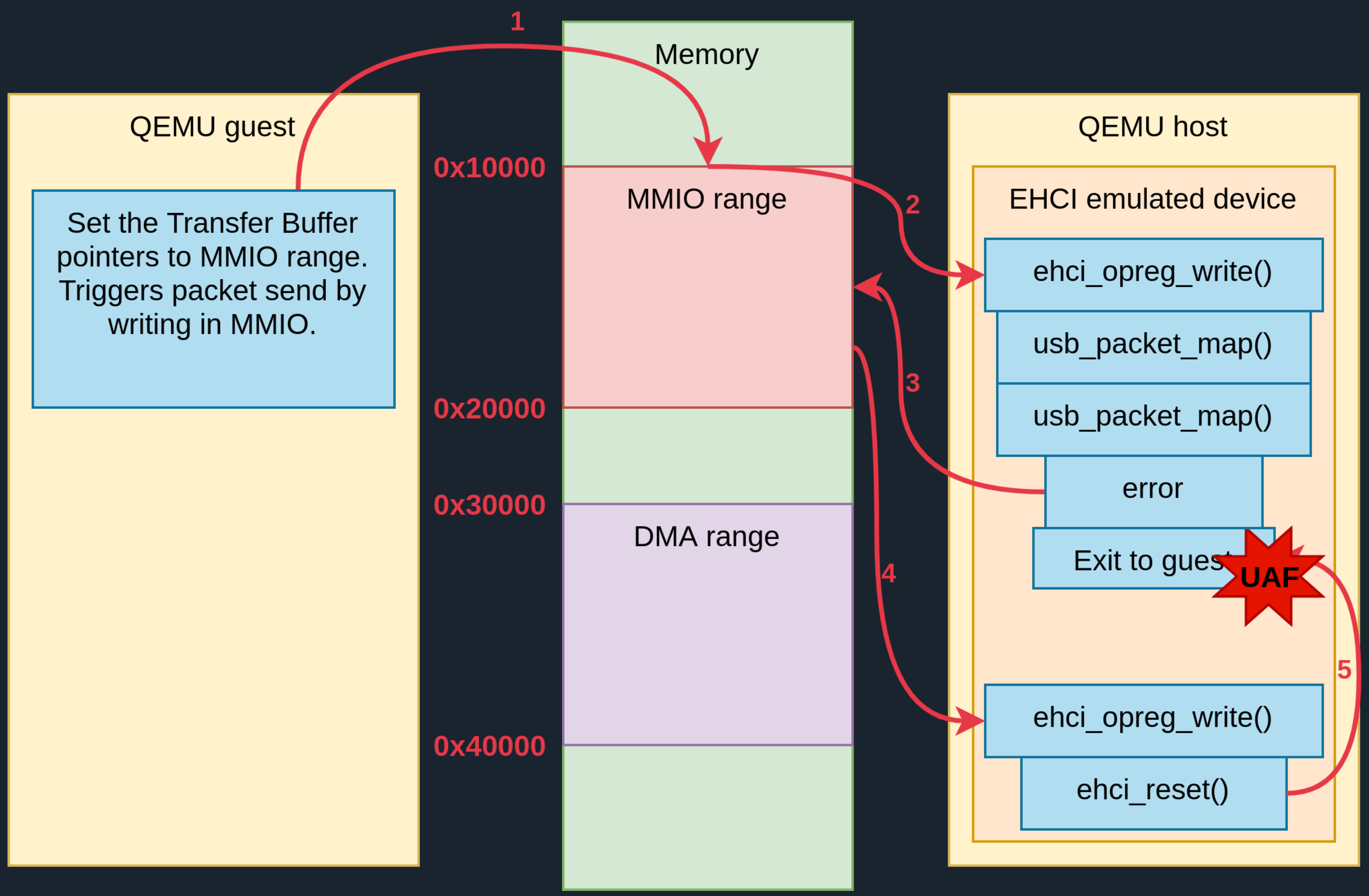
- What if the guest provides a MMIO address to the host ?
- Host will write back in the MMIO…
- Triggering another MMIO handler !
  - Depending on the callback, some device structures might be freed
- Execution is given back to the first MMIO handler…
- That might use freed objects !
  - This behavior is not vulnerable by default
  - But it is a vulnerability pattern !



Recursive MMIO

CVE-2021-3750: VME in QEMU

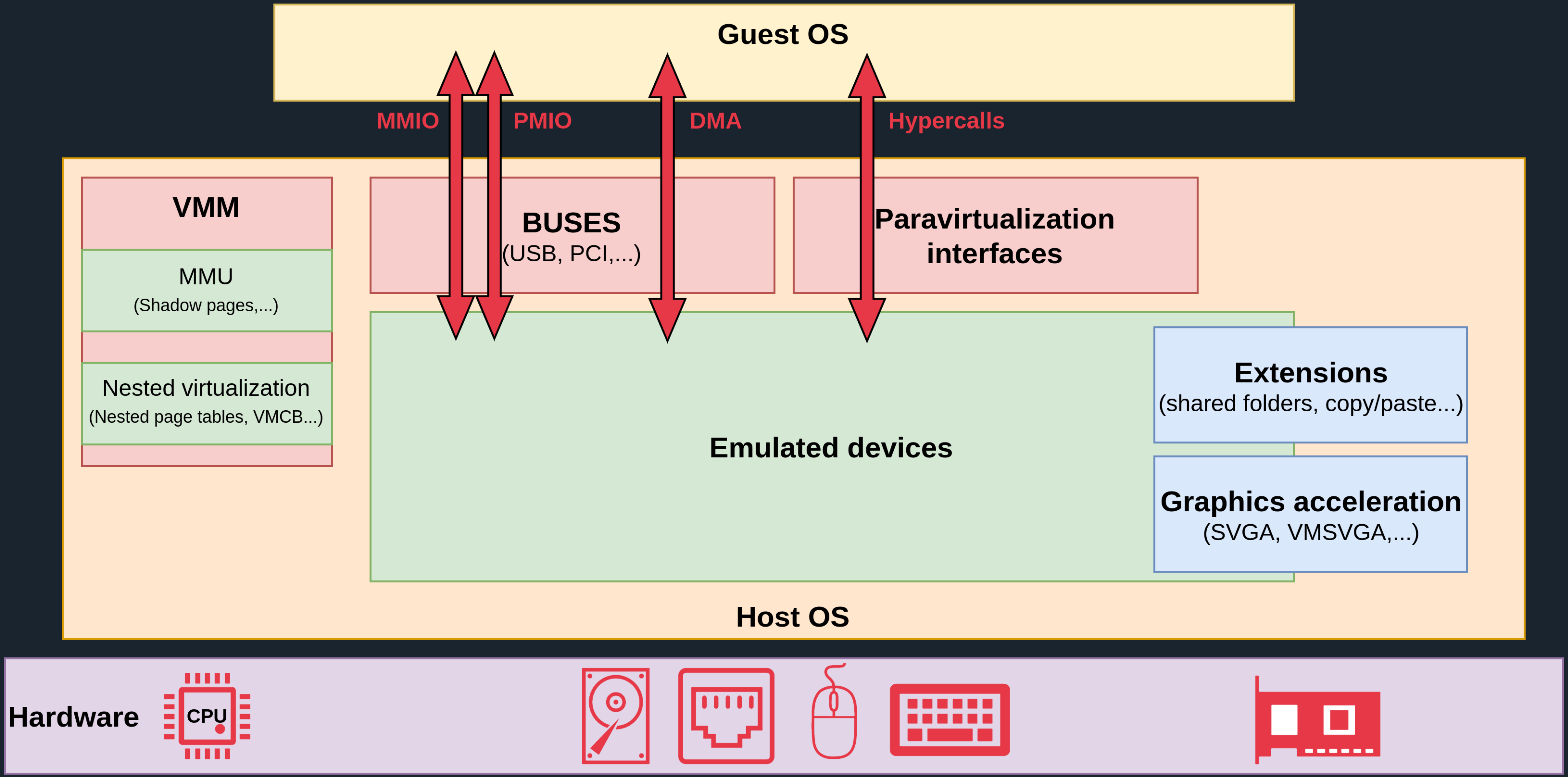
- Recursive MMIO in emulation of USB EHCI (2.0) in QEMU
- Set transfer buffer address of two first packets in MMIO region
- Trigger send packets
- QEMU tries to map the buffers
  - Fail on second buffer
  - Error handling will write in MMIO
    - Might reset the device
    - Free objects still in use



https://qiuhao.org/Matryoshka_Trap.pdf

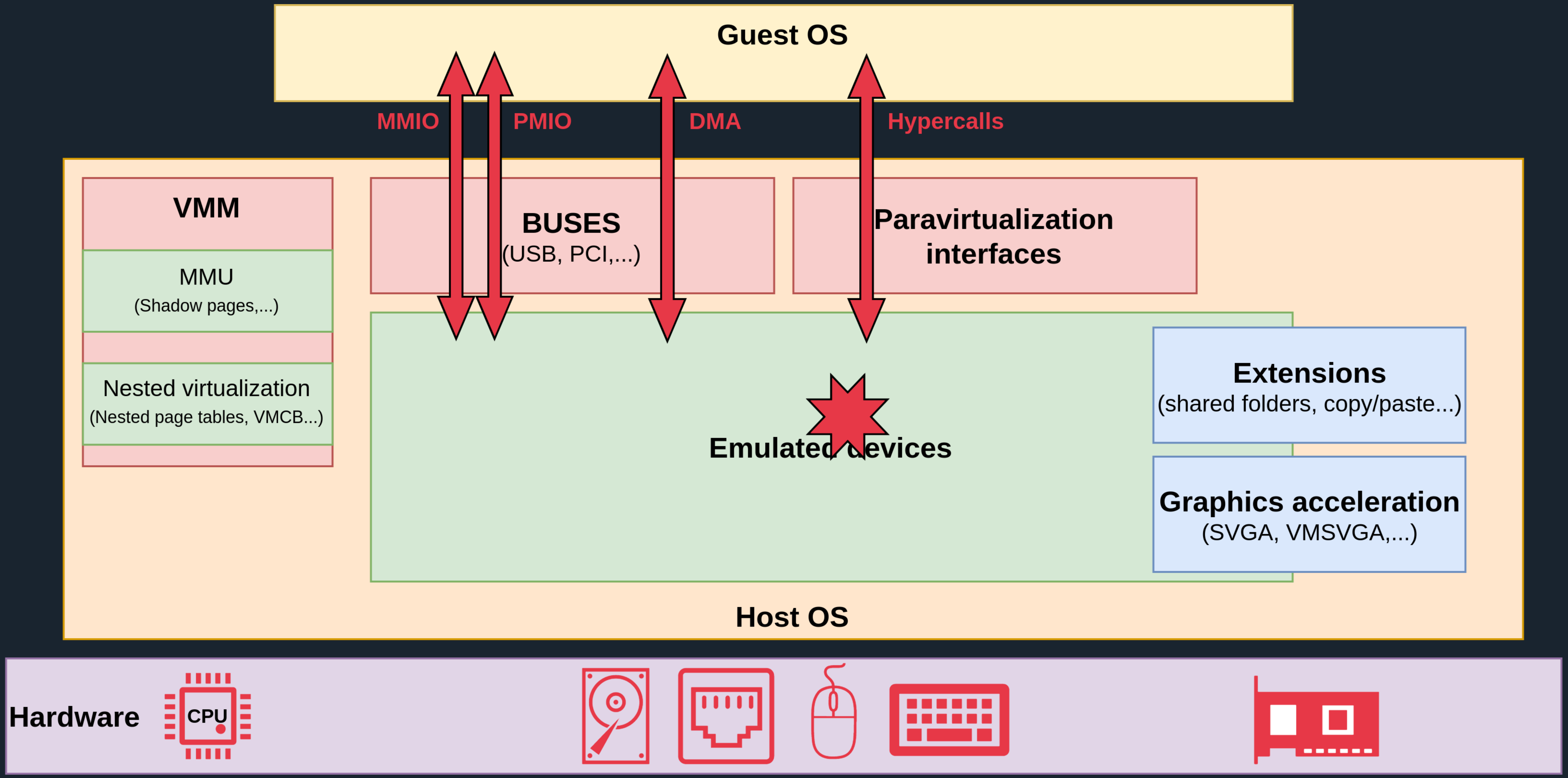
Attack surface

Hypervisor’s attack surface



Attack surface

Hypervisor's attack surface



Device Emulation bugs

CVE-2023-21987: stack buffer overflow in VirtualBox

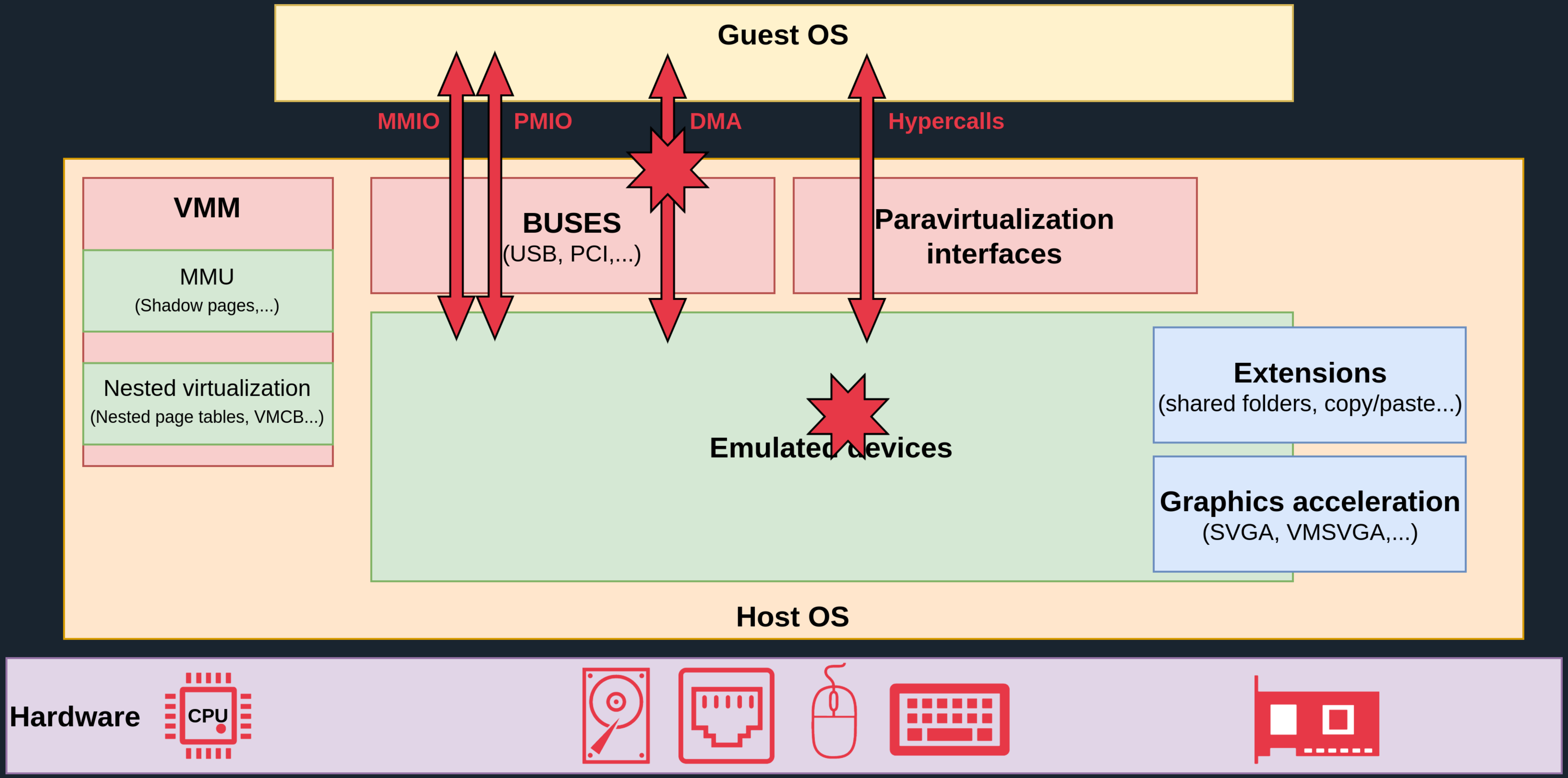
Stack buffer overflow in TPM device emulator (VirtualBox)

```
 * @param   pu64                    Where to store the read data.
 * @param   cb                      Number of bytes to read.
 */
static VBOXSTRICTRC tpmMmioFifoRead(PPDMDEVINS pDevIns, PDEVTPM pThis, PDEVTPMLOCALITY pLoc,
                                    uint8_t bLoc, uint32_t uReg, uint64_t *pu64, size_t cb)
{
    RT_NOREF(pDevIns);
    VBOXSTRICTRC rc = VINF_SUCCESS;

    /* Special path for the data buffer. */
    if (    (   (   uReg >= TPM_FIFO_LOCALITY_REG_DATA_FIFO
                && uReg < TPM_FIFO_LOCALITY_REG_DATA_FIFO + sizeof(uint32_t))
             || (   uReg >= TPM_FIFO_LOCALITY_REG_XDATA_FIFO
                 && uReg < TPM_FIFO_LOCALITY_REG_XDATA_FIFO + sizeof(uint32_t)))
        && bLoc == pThis->bLoc
        && pThis->enmState == DEVTPMSTATE_CMD_COMPLETION)
    {
        if (pThis->offCmdResp <= pThis->cbCmdResp - cb)
        {
            memcpy(pu64, &pThis->abCmdResp[pThis->offCmdResp], cb);
            pThis->offCmdResp += (uint32_t)cb;
        }
        else
            memset(pu64, 0xff, cb);
        return VINF_SUCCESS;
    }
```

Attack surface

Hypervisor's attack surface

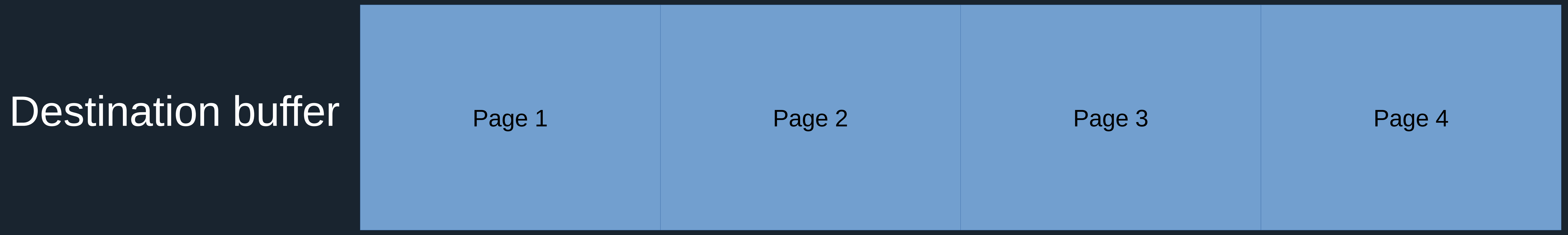


Memory management bugs

CVE-2023-21988: Uninitialized memory read in VirtualBox

- Low level API **`PGMPhysRead`** is called when doing DMA from virtual devices
- Reads guest memory page by page, goes through MMIO handlers in case of MMIO addresses

- API returns early in case of MMIO handling failure but does not set the output buffer

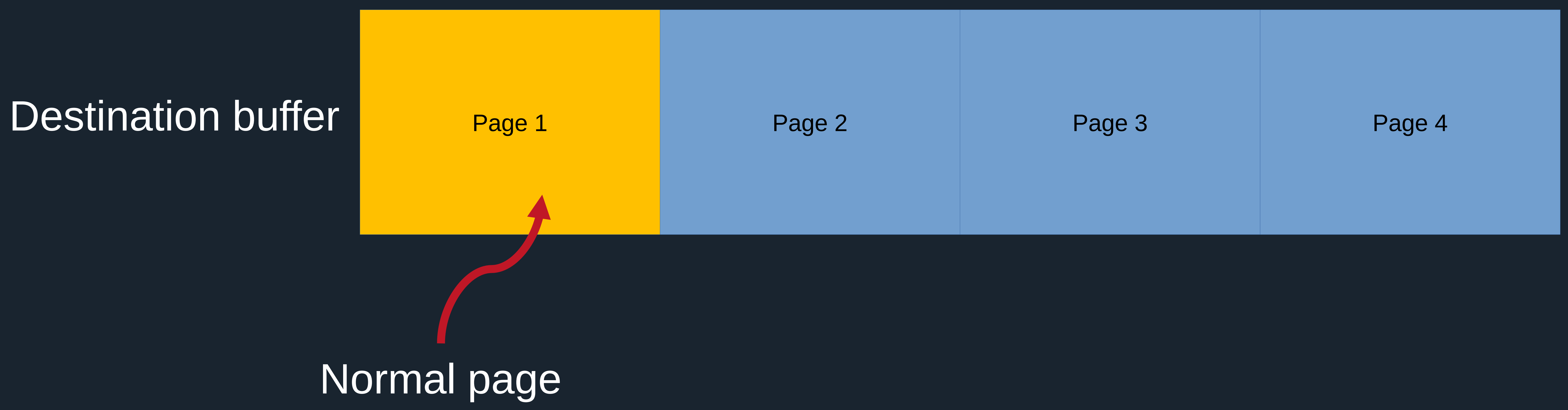


Memory management bugs

CVE-2023-21988: Uninitialized memory read in VirtualBox

- Low level API **`PGMPhysRead`** is called when doing DMA from virtual devices
- Reads guest memory page by page, goes through MMIO handlers in case of MMIO addresses

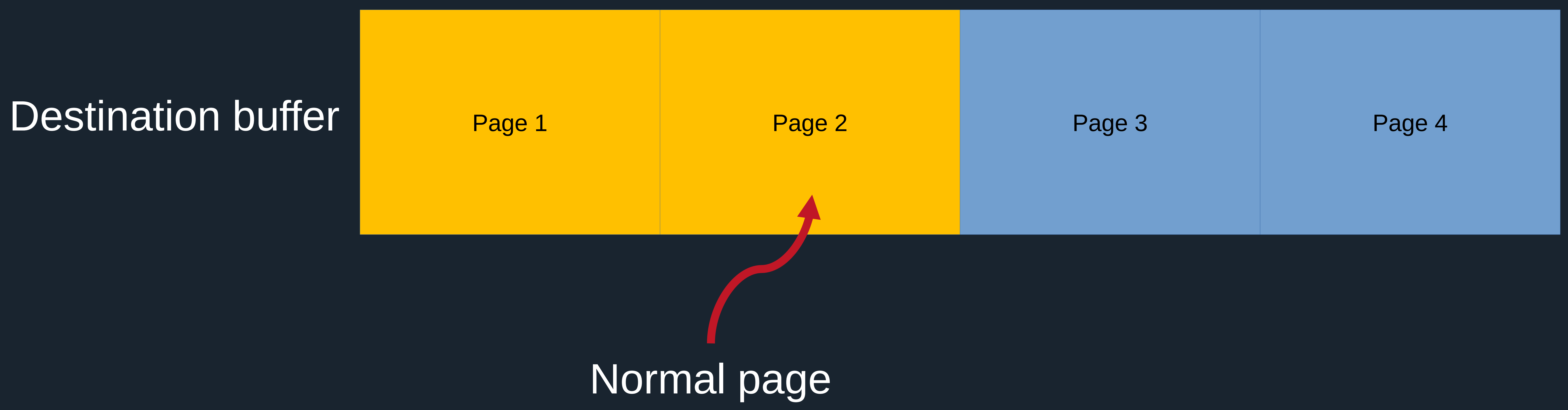
- API returns early in case of MMIO handling failure but does not set the output buffer



Memory management bugs

CVE-2023-21988: Uninitialized memory read in VirtualBox

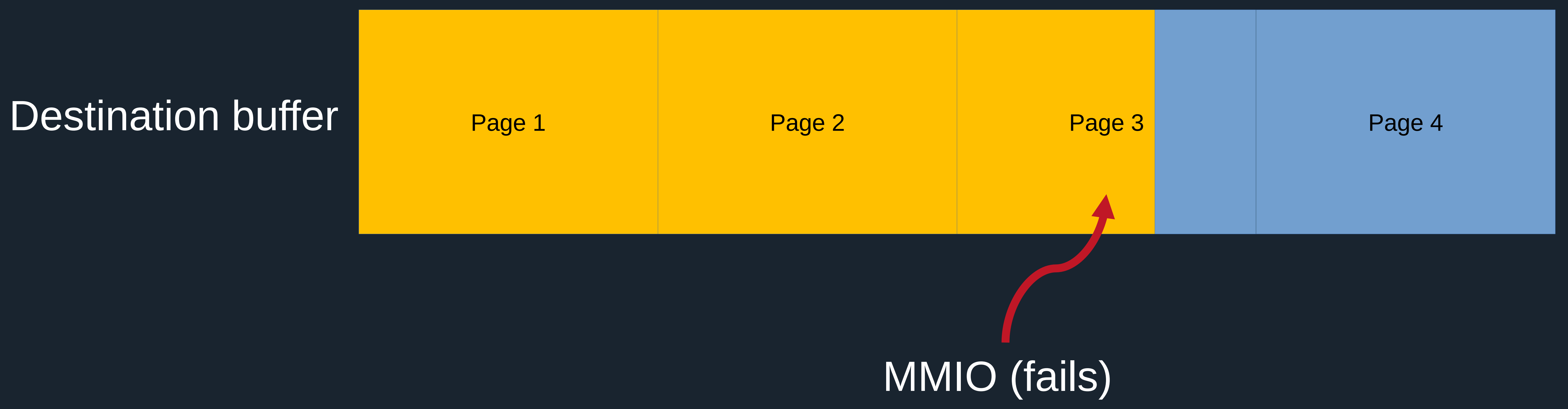
- Low level API **`PGMPhysRead`** is called when doing DMA from virtual devices
- Reads guest memory page by page, goes through MMIO handlers in case of MMIO addresses
- API returns early in case of MMIO handling failure but does not set the output buffer



Memory management bugs

CVE-2023-21988: Uninitialized memory read in VirtualBox

- Low level API **`PGMPhysRead`** is called when doing DMA from virtual devices
- Reads guest memory page by page, goes through MMIO handlers in case of MMIO addresses
- API returns early in case of MMIO handling failure but does not set the output buffer

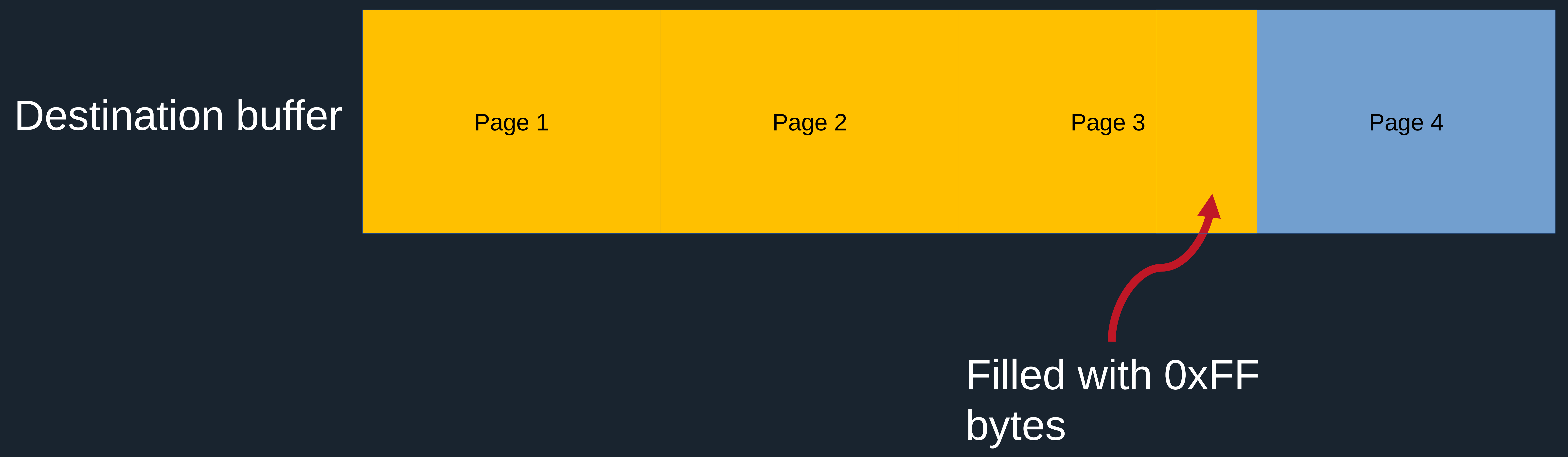


Memory management bugs

CVE-2023-21988: Uninitialized memory read in VirtualBox

- Low level API **`PGMPhysRead`** is called when doing DMA from virtual devices
- Reads guest memory page by page, goes through MMIO handlers in case of MMIO addresses

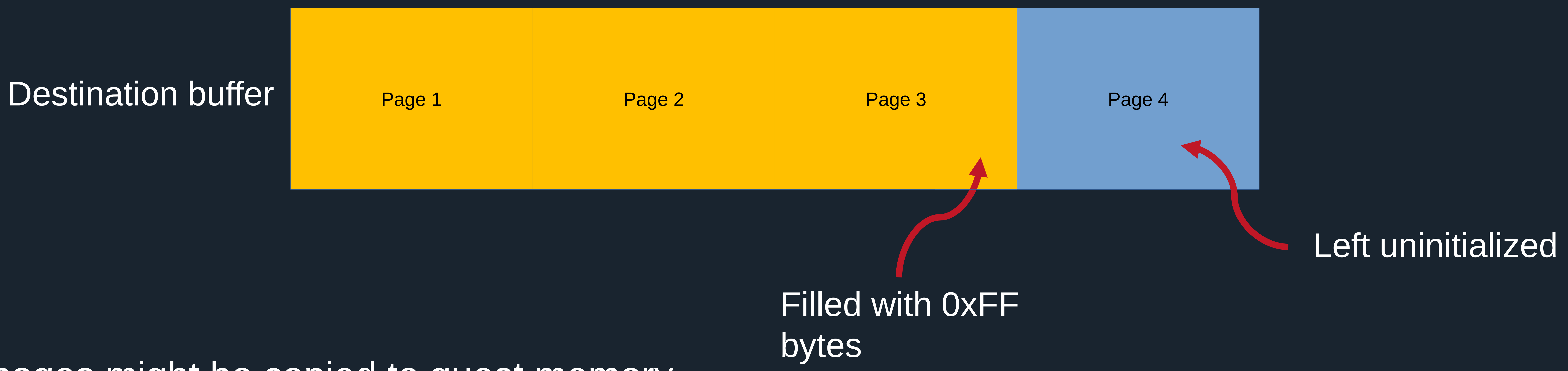
- API returns early in case of MMIO handling failure but does not set the output buffer



Memory management bugs

CVE-2023-21988: Uninitialized memory read in VirtualBox

- Low level API **`PGMPhysRead`** is called when doing DMA from virtual devices
- Reads guest memory page by page, goes through MMIO handlers in case of MMIO addresses
- API returns early in case of MMIO handling failure but does not set the output buffer



- All 4 pages might be copied to guest memory…
  - …leaking uninitialized data to guest !

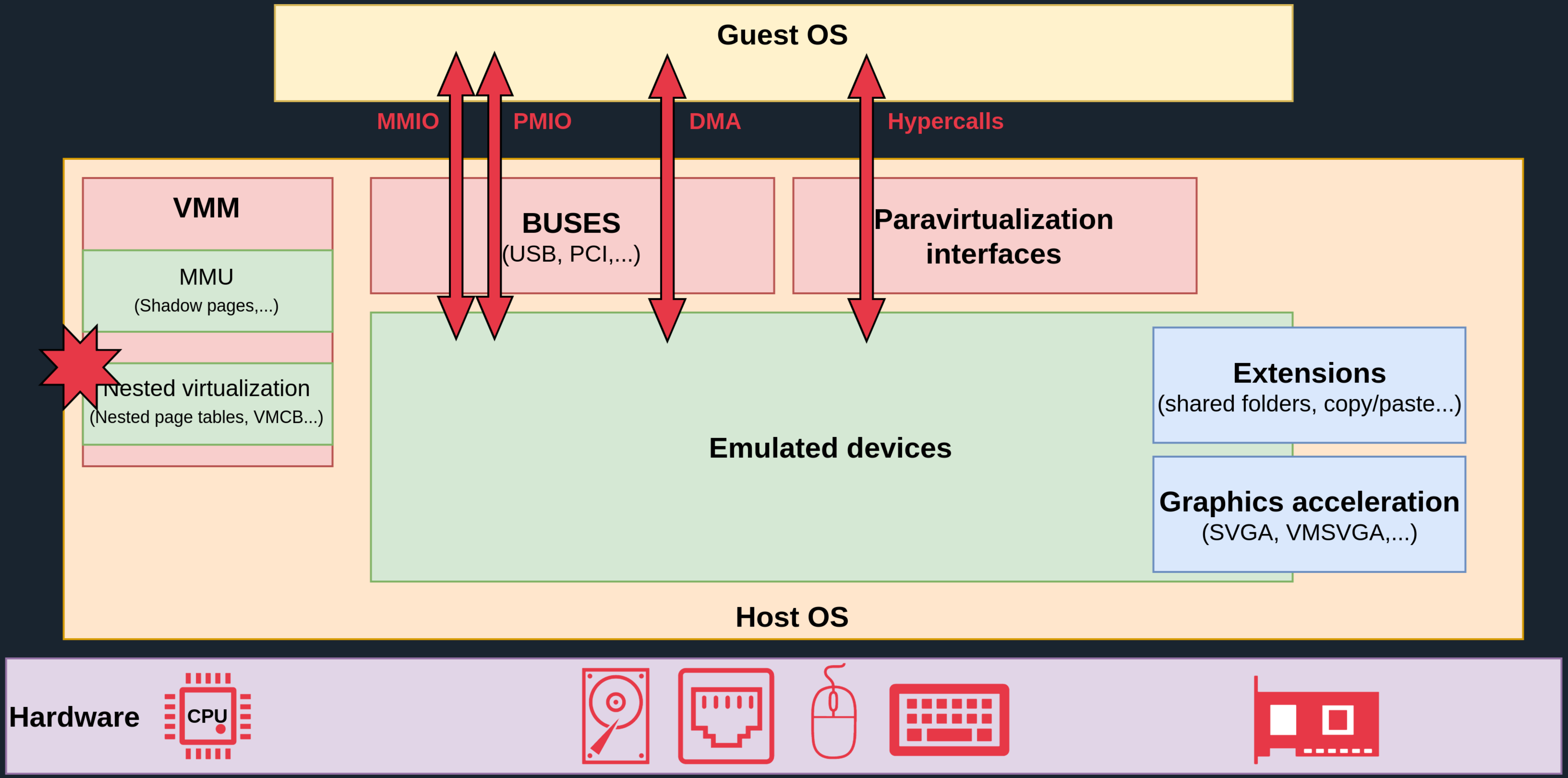
Device Emulation bugs

Escaping from VirtualBox at Pwn2Own Vancouver 2023

- Uninitialized memory bug can be used to leak either heap or stack data
  - Can be used to break ASLR, leak eventual stack canaries…
- Chain with the TPM stack buffer overflow
  - Overwrite the return address and build a ROP-chain
  - Get code execution on host OS
- 100% reliable VM escape from VirtualBox!

Attack surface

Hypervisor’s attack surface



Nested Virtualization bugs

CVE-2021-29657: Arbitrary host MSR access in QEMU

- Nested virtualization is handled by QEMU
  - When creating a nested VM, the hypervisor needs to check the values of the configuration structure

Nested Virtualization bugs

CVE-2021-29657: Arbitrary host MSR access in QEMU

- Nested virtualization is handled by QEMU
  - When creating a nested VM, the hypervisor needs to check the values of the configuration structure

- Double fetch in nested VMCB configuration
  - First fetch validates the configuration
  - Second fetch sets the actual configuration in the hypervisor

Nested Virtualization bugs

CVE-2021-29657: Arbitrary host MSR access in QEMU

- Nested virtualization is handled by QEMU
  - When creating a nested VM, the hypervisor needs to check the values of the configuration structure

- Double fetch in nested VMCB configuration
  - First fetch validates the configuration
  - Second fetch sets the actual configuration in the hypervisor
  - Guest can change the data in between!

- Primitive gives access to host **MSR** registers through the guest

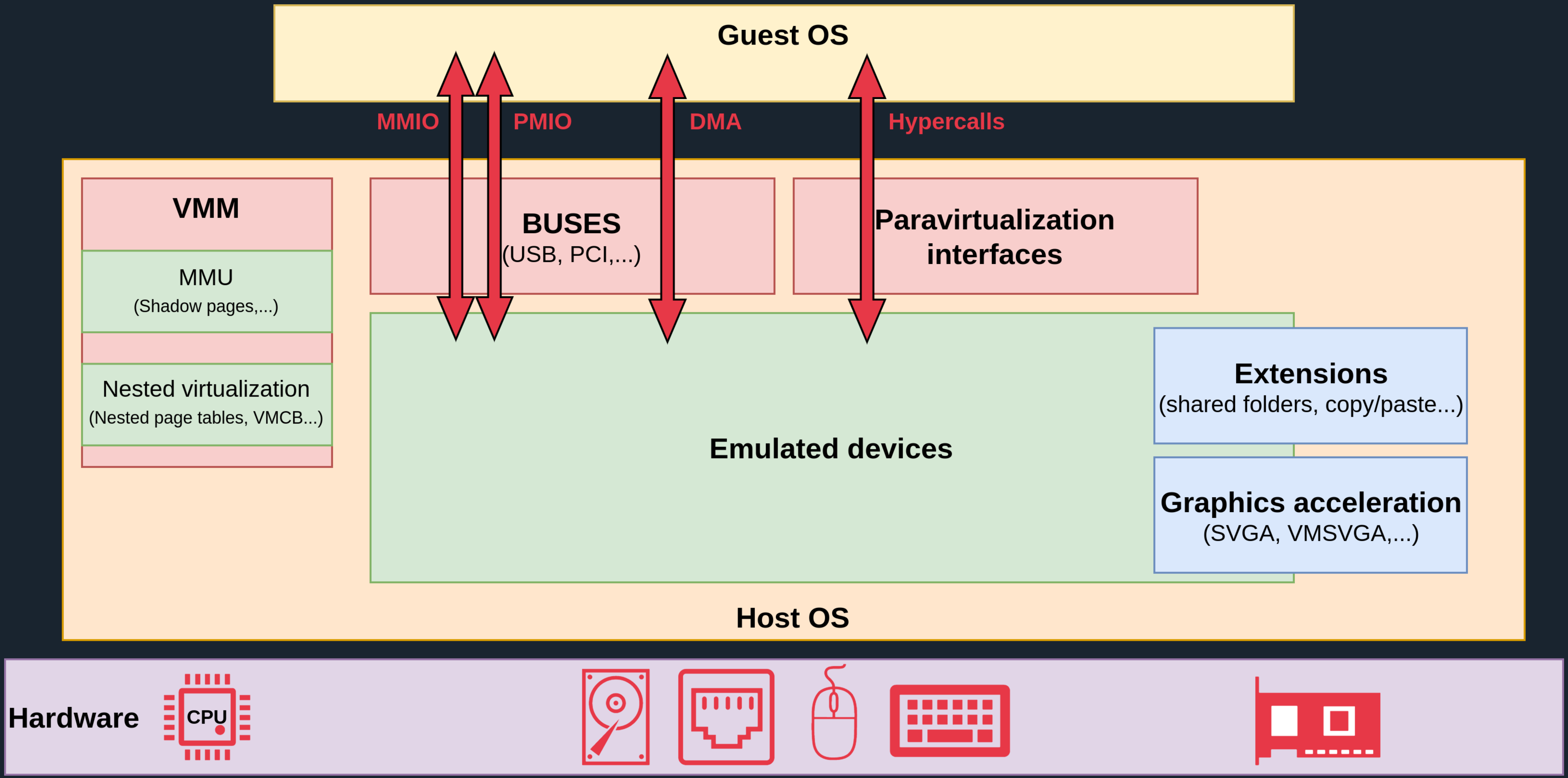
Nested Virtualization bugs

CVE-2021-29657: Arbitrary host MSR access in QEMU

- Nested virtualization is handled by QEMU
  - When creating a nested VM, the hypervisor needs to check the values of the configuration structure
- Double fetch in nested VMCB configuration
  - First fetch validates the configuration
  - Second fetch sets the actual configuration in the hypervisor
  - Guest can change the data in between!
- Primitive gives access to host **MSR** registers through the guest
- Exploitable, but not trivial (found and exploited by Felix Wilhelm, Project Zero)
- Only affects hosts using AMD CPUs

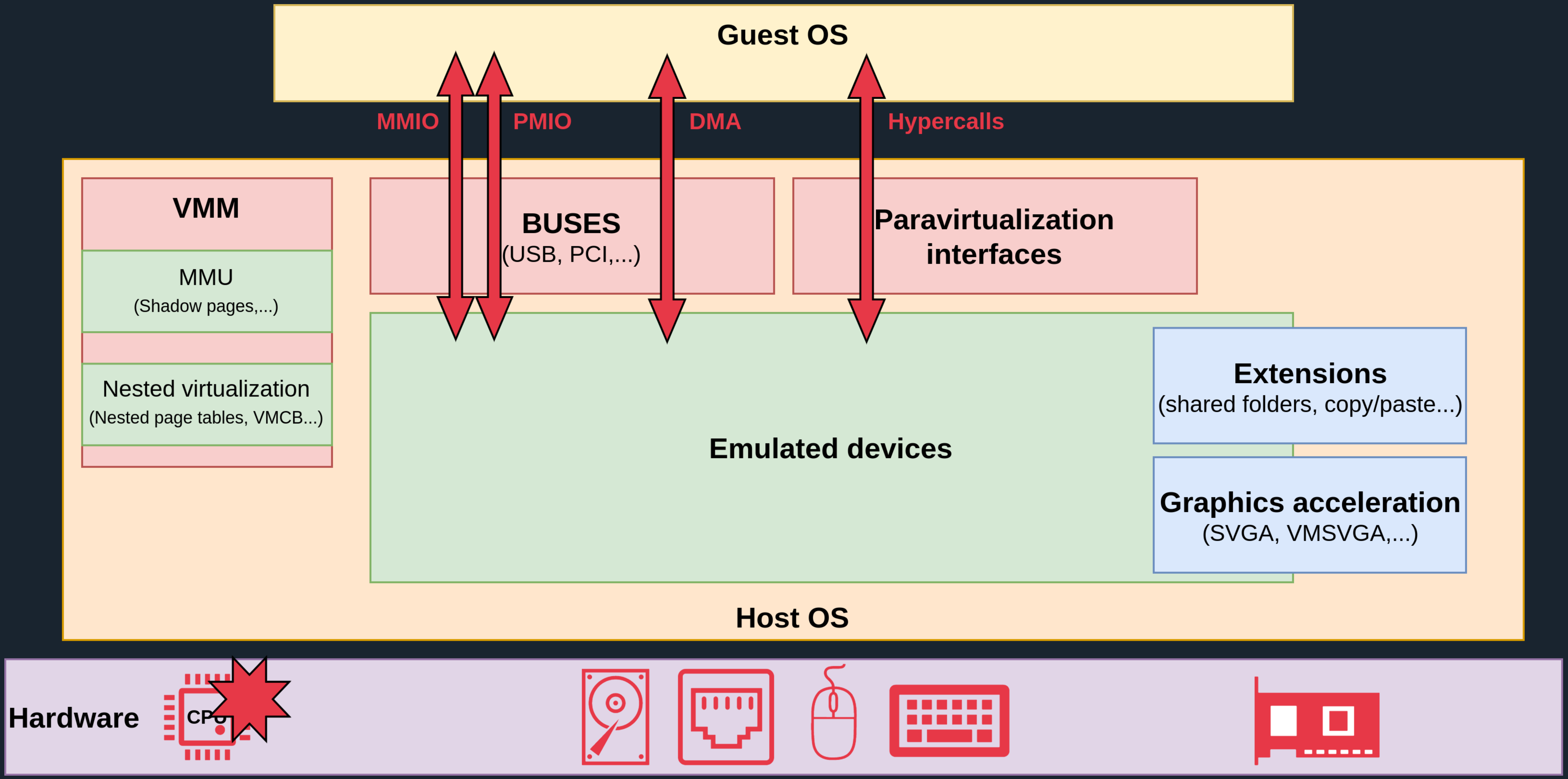
Attack surface

Hypervisor's attack surface



Attack surface

Hypervisor's attack surface



Hardware bugs ?

CPUs can’t be trusted

- CPUs are not immune to bugs
  - Some of them can be exploited from the guest
  - Can break host/guest or inter-vm isolation
- A few examples:
  - Meltdown/Spectre
  - CVE 2018-3646: L1 Terminal Fault (L1TF)
- Root causes are often due to performance features
  - The fix often has a performance trade-off

Conclusion

Conclusion

- Hypervisors expose a wide and complex attack surface
- Most disclosed vulnerabilities still reside in emulated / paravirtualized devices
    - Simpler to approach for an attacker
    - You don’t have to understand everything about virtualization to hunt bugs here
- But the context still exposes some very specific vulnerabilities
    - Recursive MMIO (or DMA Reentrancy)
    - Weird emulation bugs
    - Hardware bugs
- This was a short introduction
    - There is much more to say on the subject

Conclusion

- Try it yourself !
- Fun and lucrative
  - Microsoft Hyper-V's bounty program awards

Vulnerability Type	Functioning Exploit	Report Quality	Payout range (USD)*
RCE	Yes	High	$250,000
	No	High	$200,000
	No	Low	$50,000

  - Pwn2Own

Target	Prize	Master of Pwn Points	Eligible for Add-on Prize
Oracle VirtualBox	$40,000	4	Yes
VMware Workstation	$80,000	8	Yes
VMware ESXi	$150,000	15	No
Microsoft Hyper-V Client	$250,000	25	Yes

A few links

- https://docs.saferwall.com/blog/virtualization-internals-part-1-intro-to-virtualization
- https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf
- https://qiuhao.org/Matryoshka_Trap.pdf
- https://alisa.sh/slides/HypervisorVulnerabilityResearch2020.pdf
- https://www.synacktiv.com/sites/default/files/2023-10/hexacon_breaking_out_of_the_box.pdf
- https://www.synacktiv.com/sites/default/files/2020-10/Speedpwning_VMware_Workstation.pdf
- https://github.com/shogunlab/awesome-hyper-v-exploitation
- https://www.keysight.com/blogs/tech/nwvs/2023/02/24/remote-code-execution-with-esxi-cve-2021-21974-vmware-esxi-heap-overflow
- https://googleprojectzero.blogspot.com/2021/06/an-epyc-escape-case-study-of-kvm.html

?

SYNACKTIV

www.linkedin.com/company/synacktiv

www.twitter.com/synacktiv

www.synacktiv.com