A Drop-In Sandbox Solution for Private Data Processing in Untrusted Confidential Virtual Machines

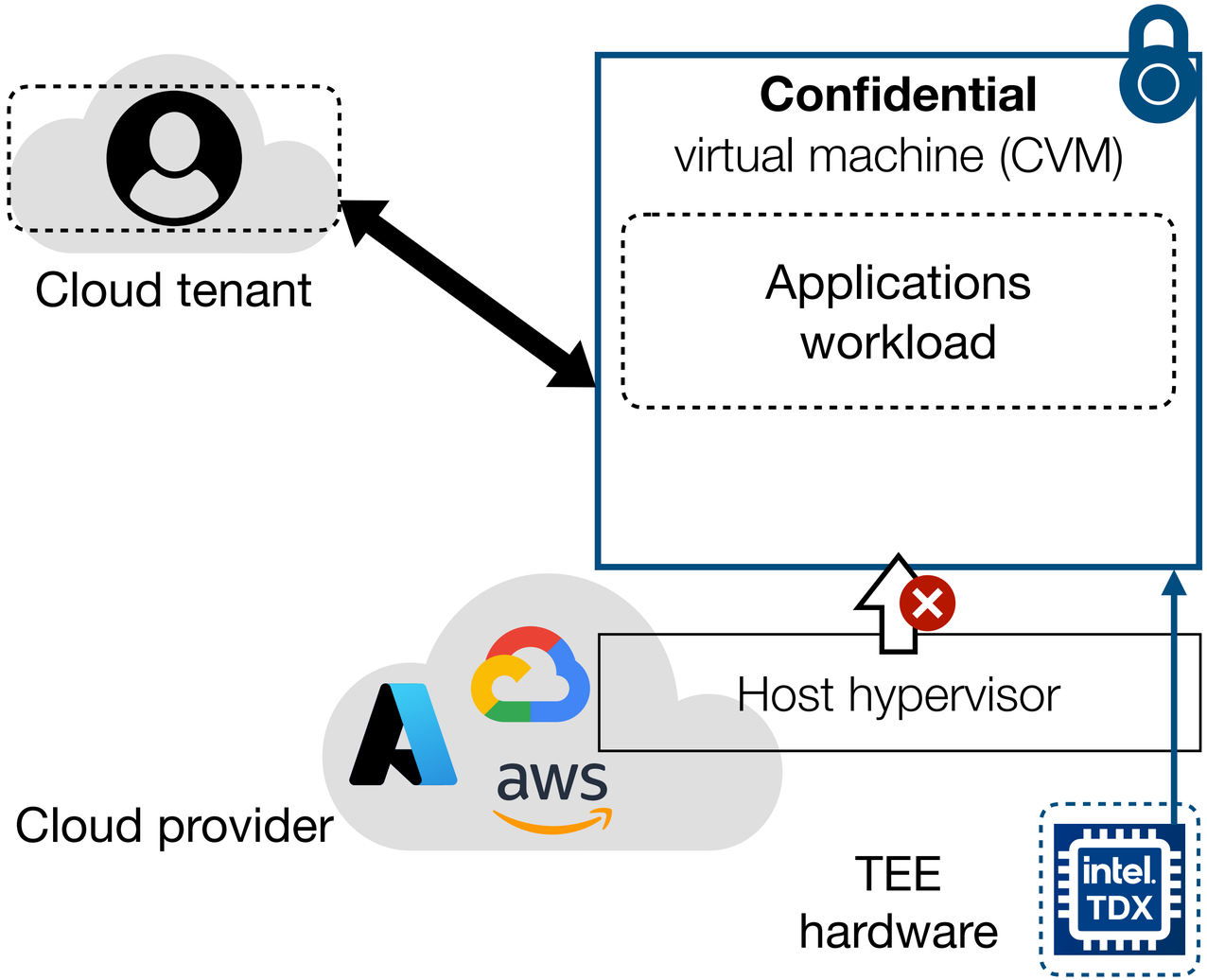
Chuqi Zhang[1,2], Rahul Priolkar[2], Yuancheng Jiang[1], Yuan Xiao[3,4], Mona Vij[4], Zhenkai Liang[1], Adil Ahmad[2]

Rotterdam, Netherlands, EuroSys 2025

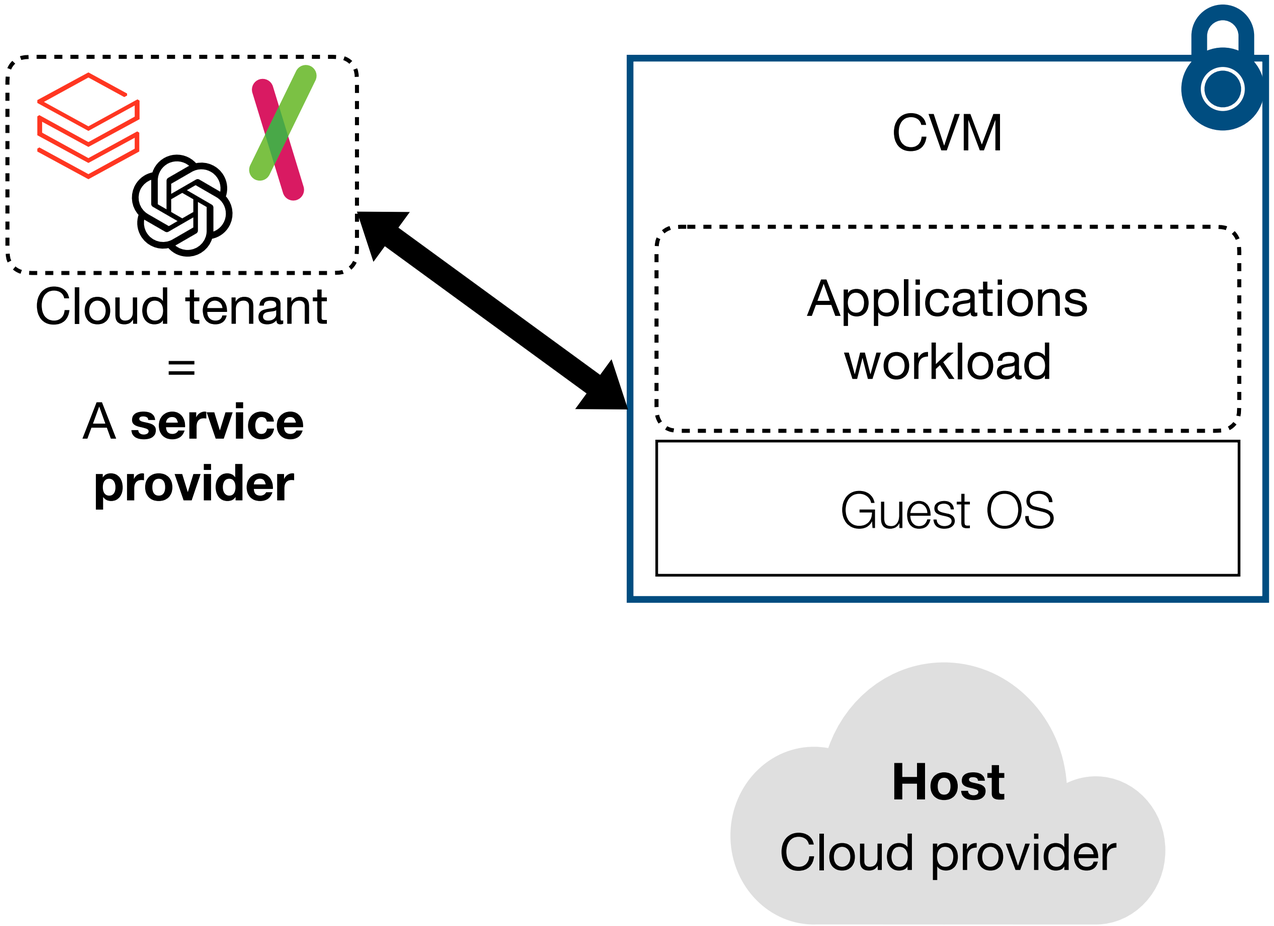
EREBOR

NUS[1] National University of Singapore
Arizona State University[2]
上海科技大学[3] ShanghaiTech University
intel labs[4]

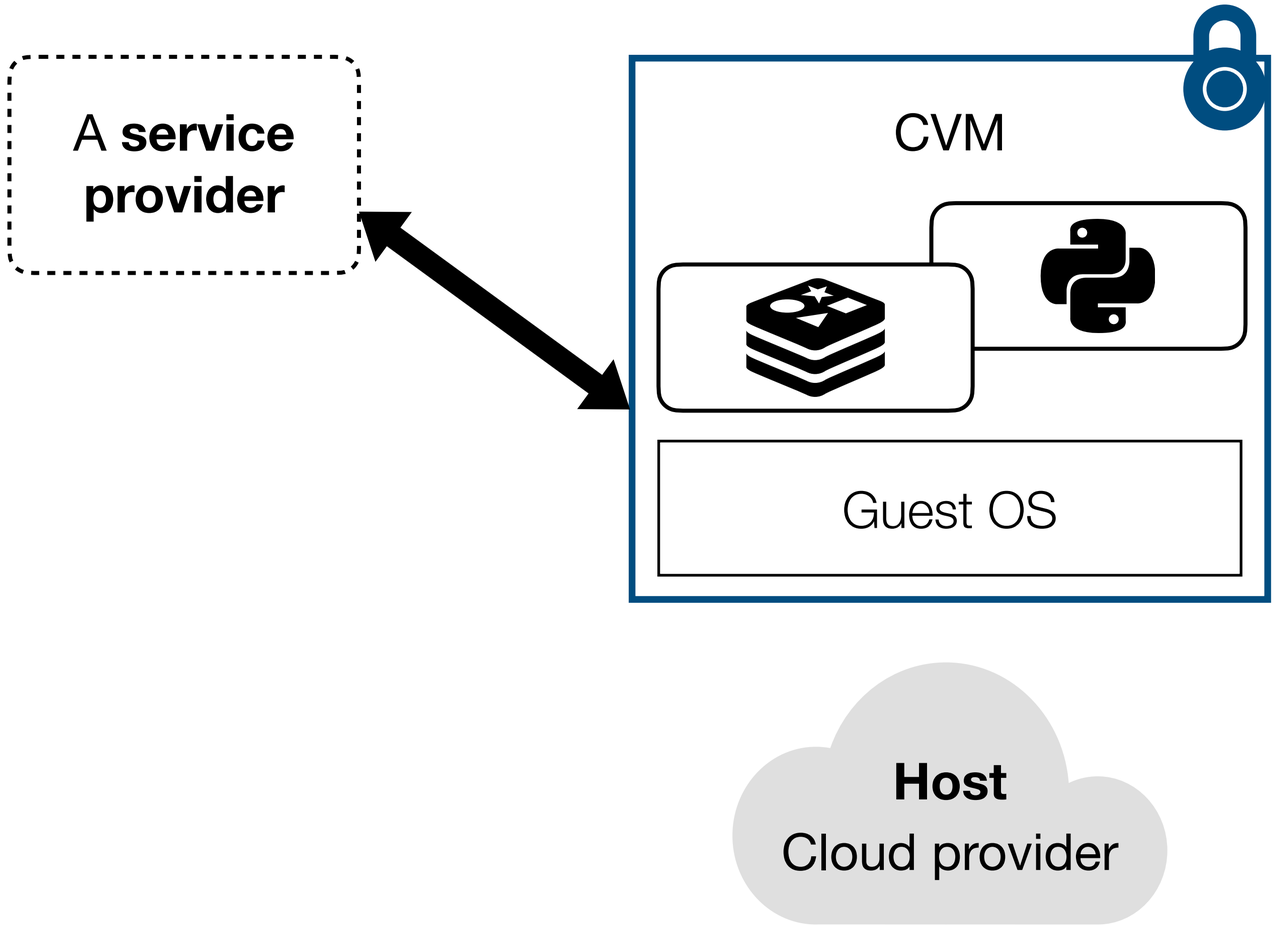
Confidential VMs are emerging in cloud computing



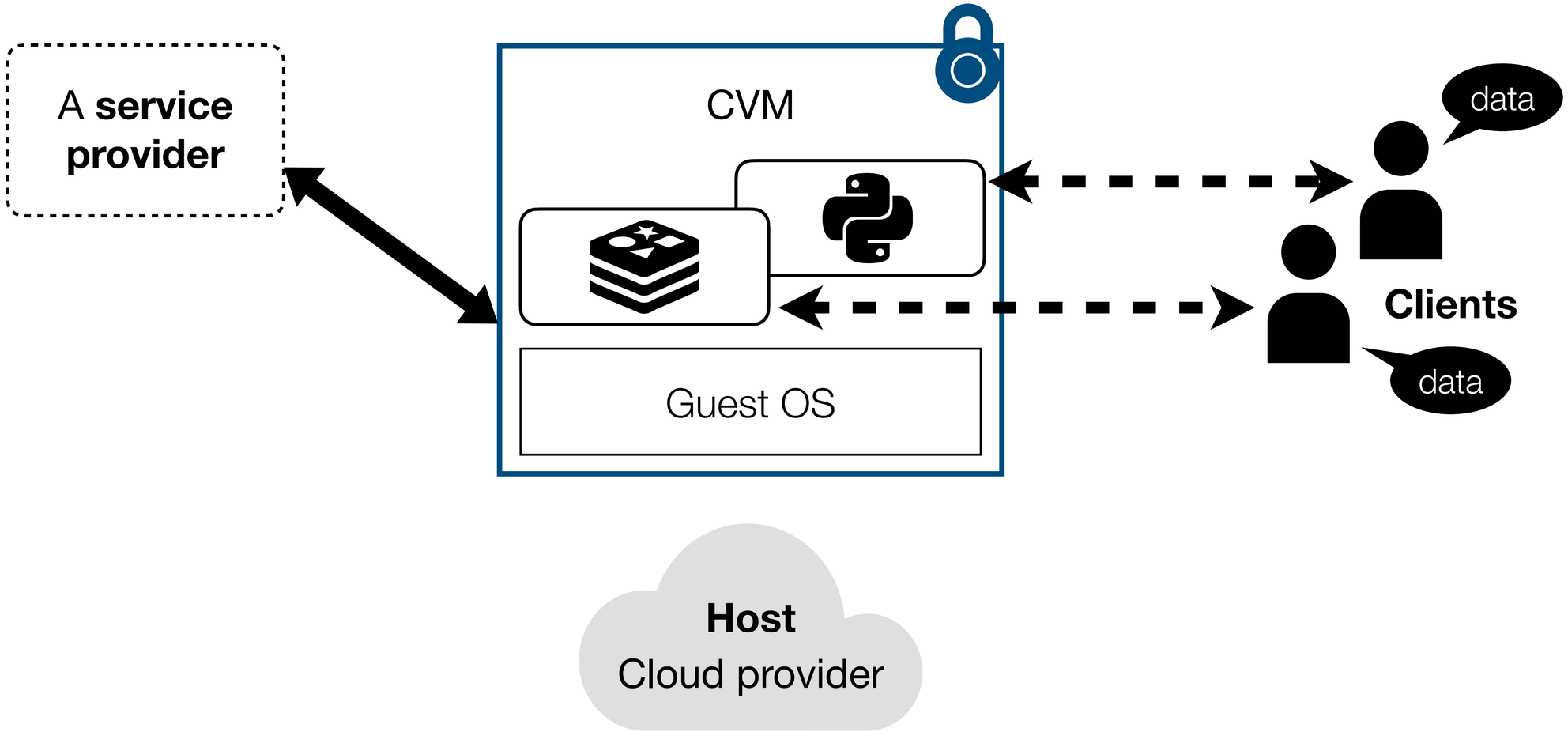
Software-as-a-service (SaaS) in CVM



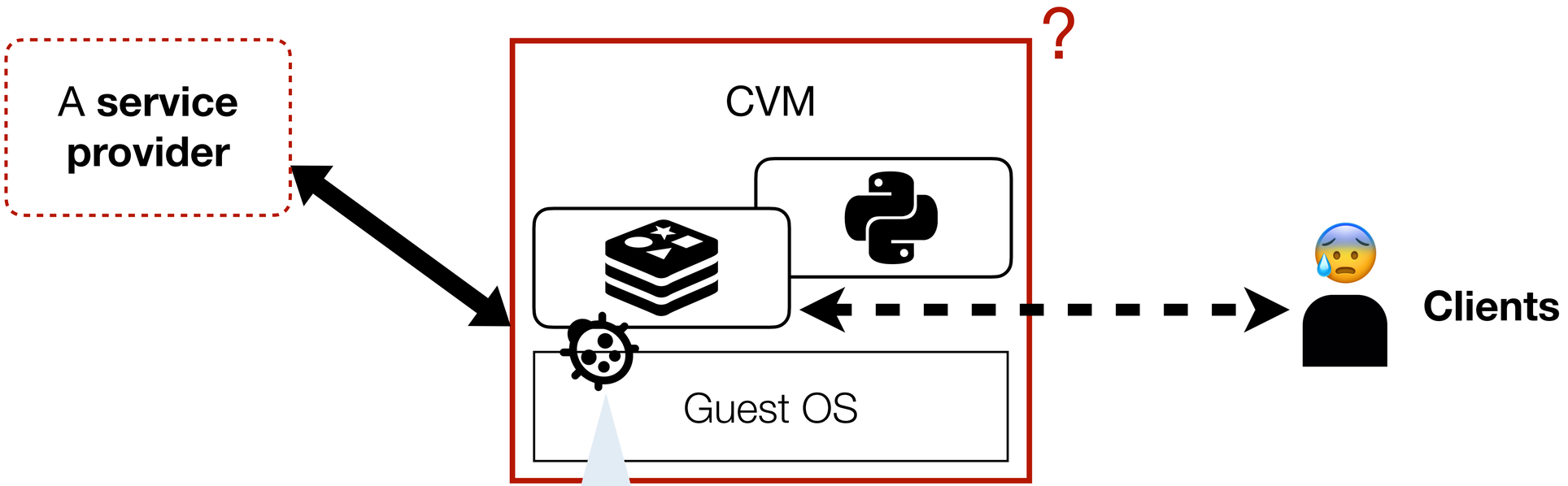
Software-as-a-service (SaaS) in CVM



Software-as-a-service (SaaS) in CVM



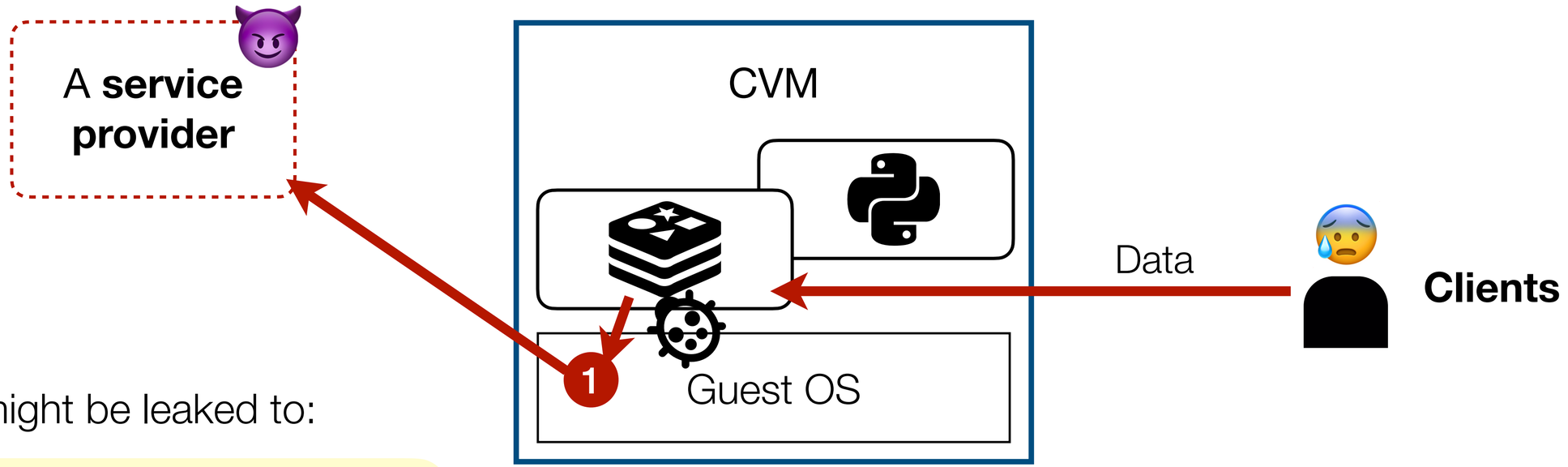
Problem: client data leakage in CVM-SaaS



Clients' view: the CVM software stack **is untrusted.**

Buggy large codebase. or Curious to collect data.

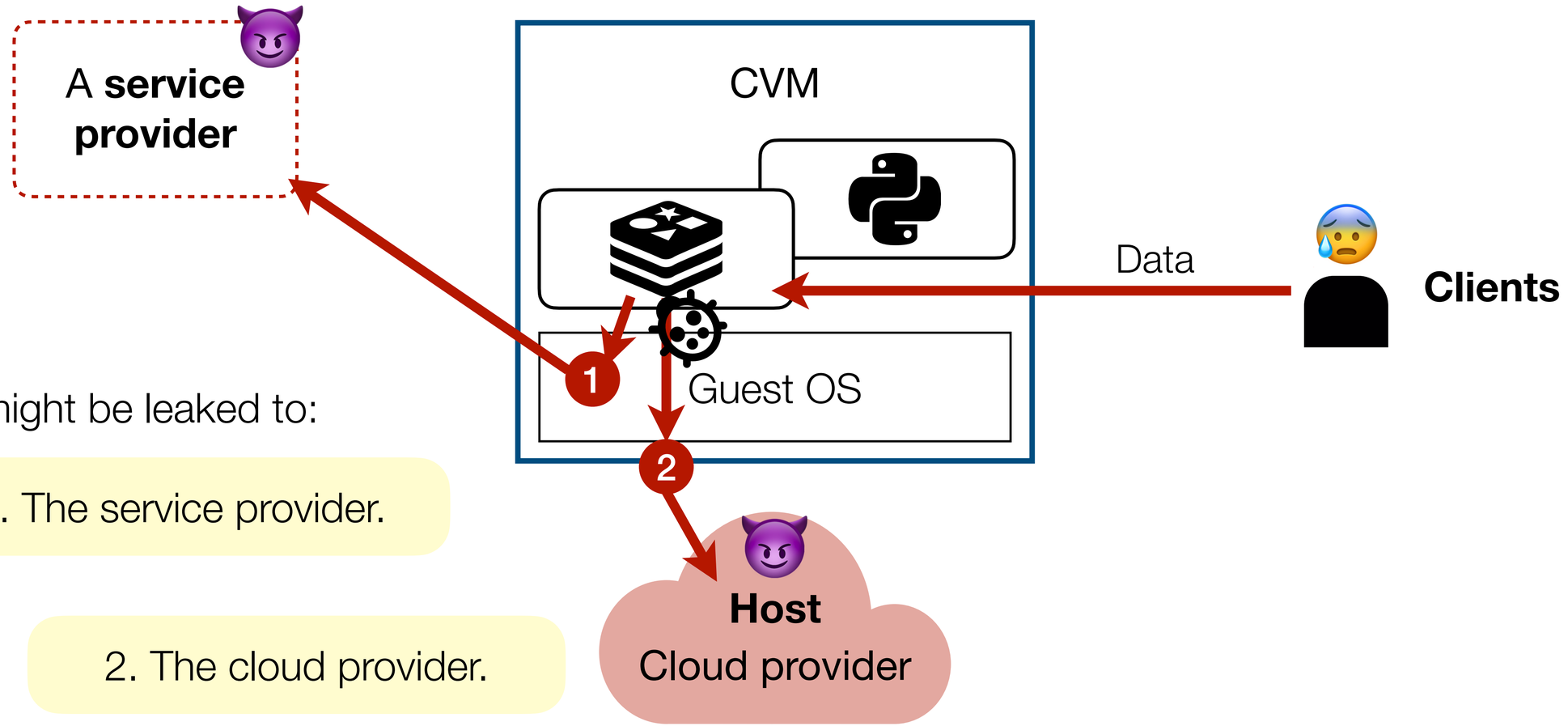
Potential data leakage vectors in CVM-SaaS



Data might be leaked to:

1. The service provider.

Potential data leakage vectors in CVM-SaaS

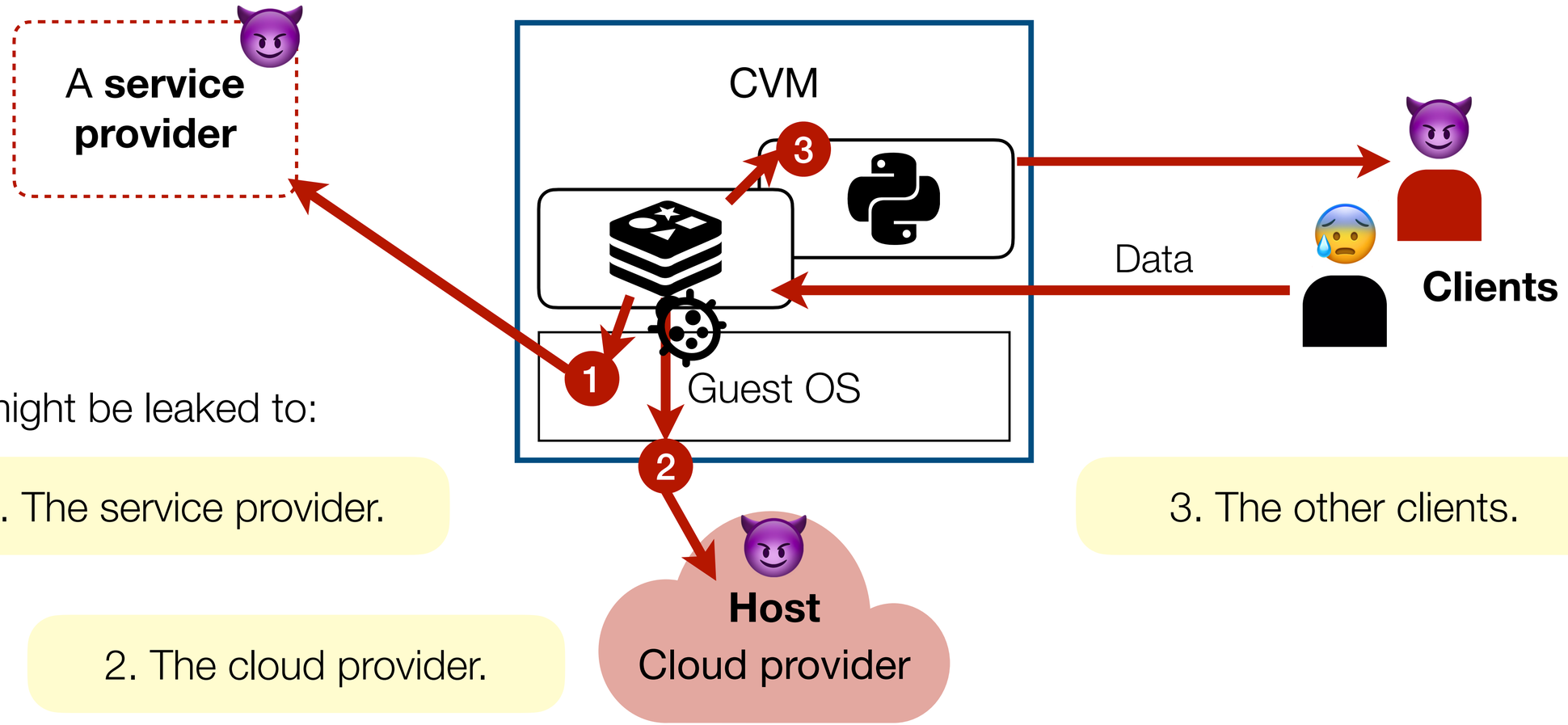


Data might be leaked to:

1. The service provider.

2. The cloud provider.

Potential data leakage vectors in CVM-SaaS



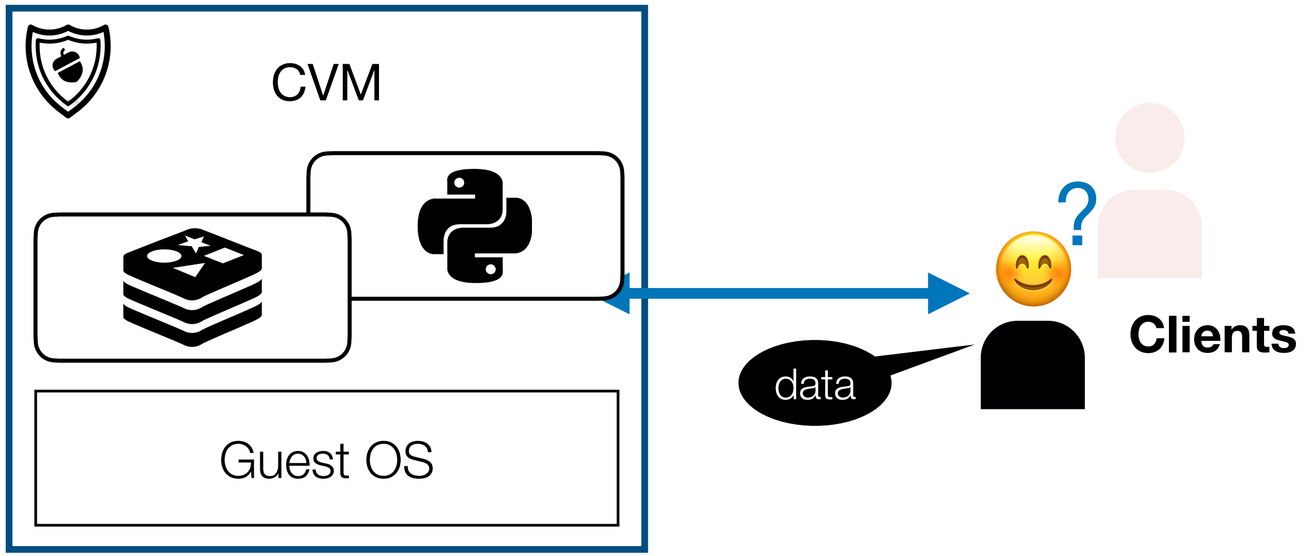
Data might be leaked to:

1. The service provider.

2. The cloud provider.

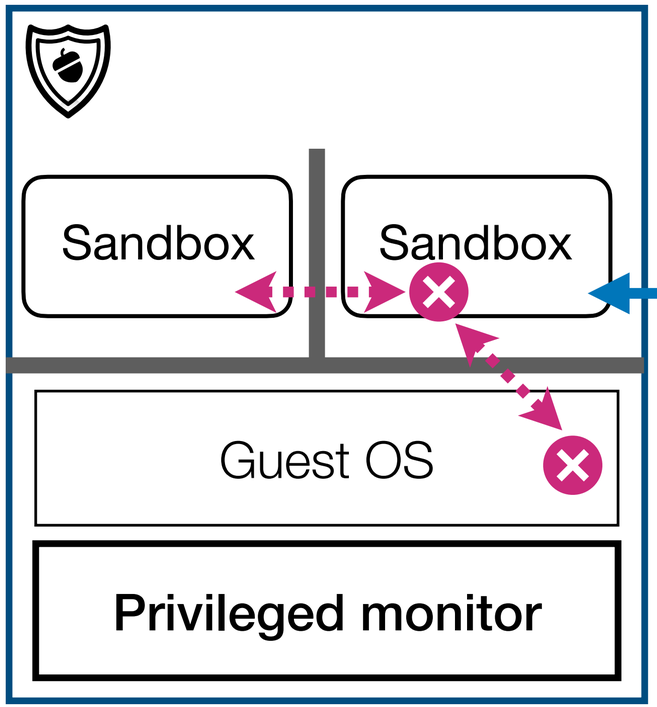
3. The other clients.

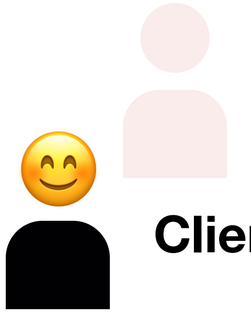
How to protect client data privacy in CVM-SaaS?



EREBOR overview: intra-CVM data sandboxing

✗ Confined sandbox container environment.





Clients

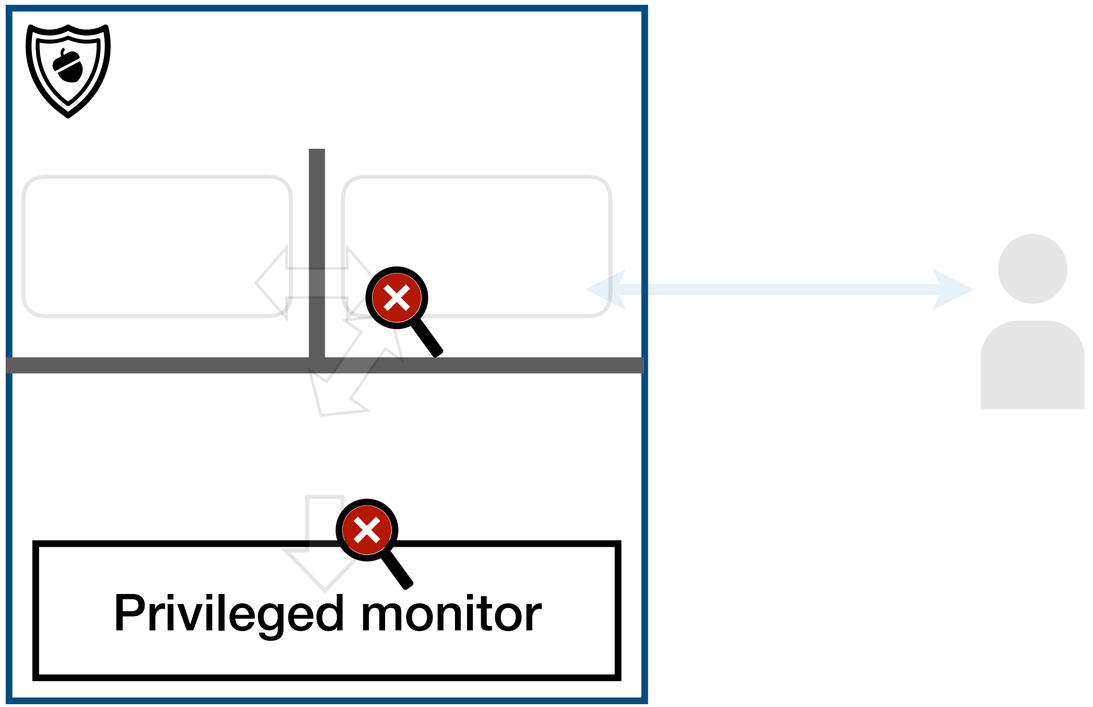
✓ Data: only accessible in sandbox.

√ Tiny, trustworthy computing base.

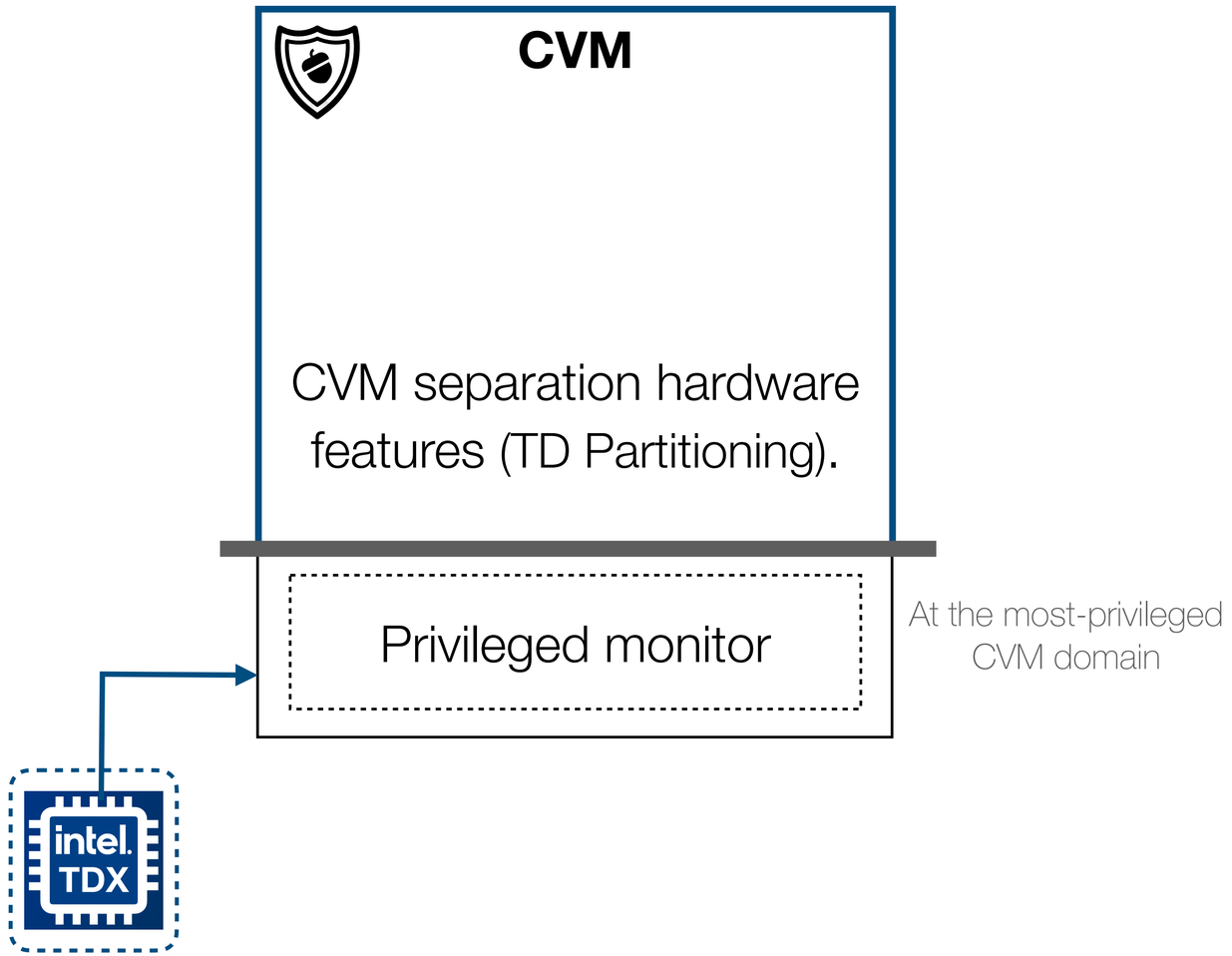
Research question 1: In-CVM privileged monitor

In-CVM privileged monitor

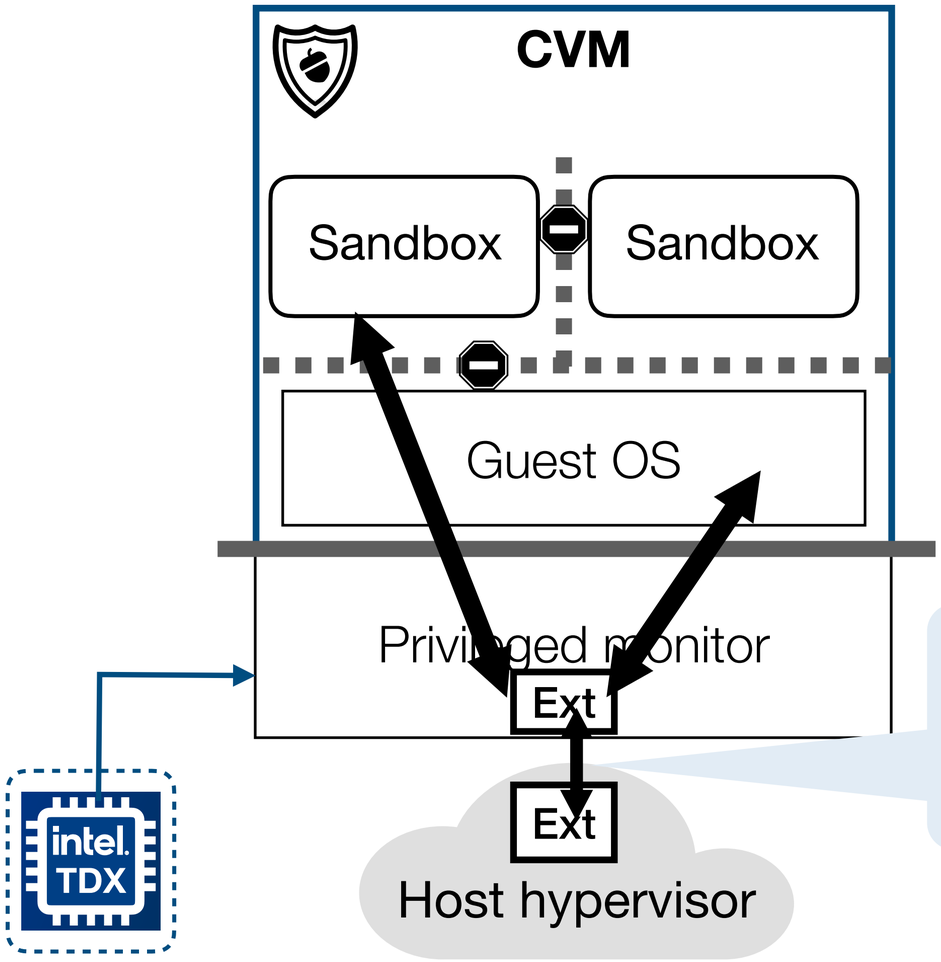
How to design it?



Prior solution: hardware-enabled CVM separation

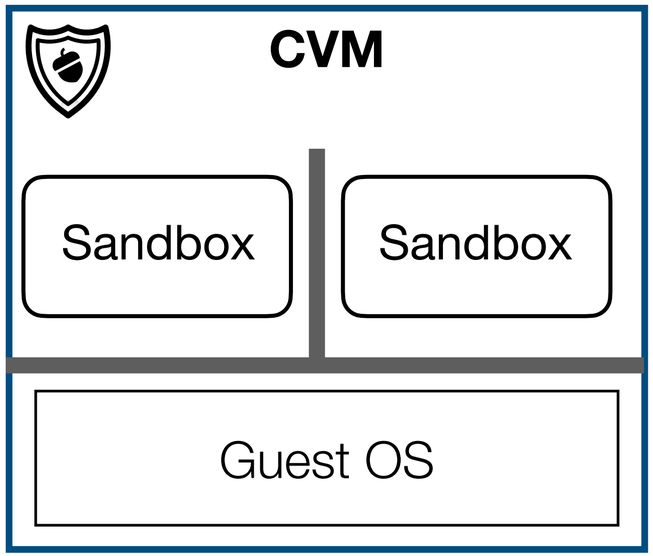


Prior solution: hardware-enabled CVM separation

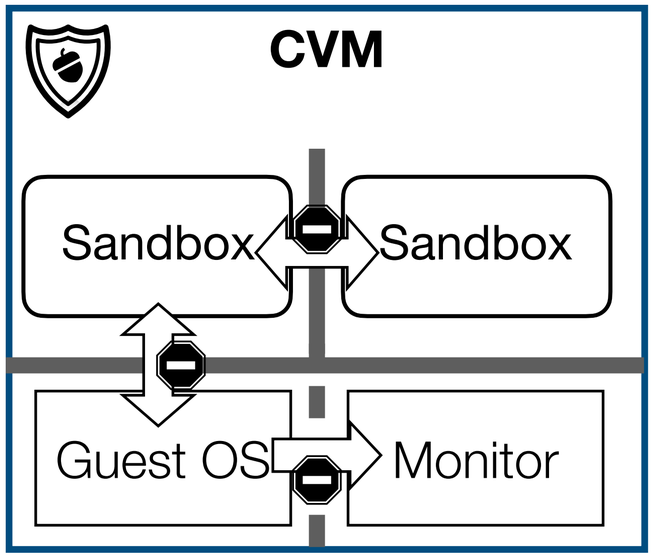


Problem of deployment:
Need **hypervisor change** to help **intra-CVM isolation**.

We choose to design a monitor by intra-kernel isolation



We choose to design a monitor by intra-kernel isolation



At the kernel (supervisor) hardware privileged level

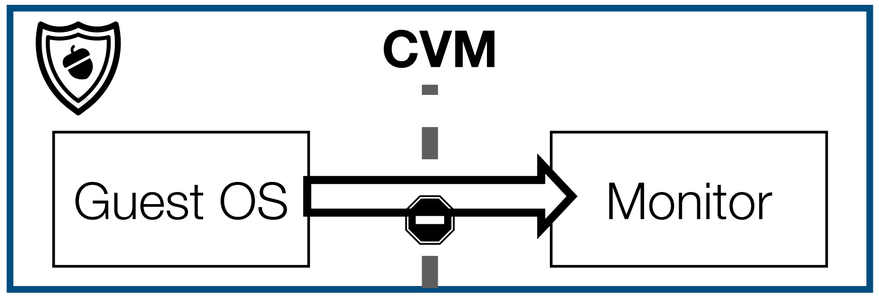
Drop-in deployment

No cloud infrastructure changes required.

Host

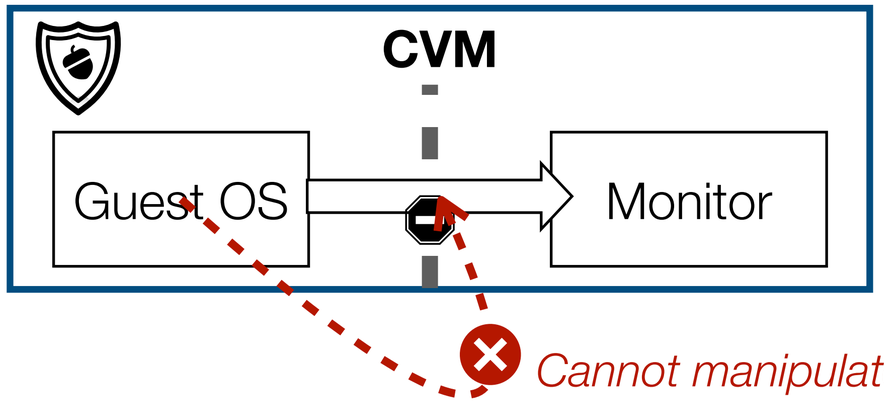
Cloud provider

Key requirements for intra-kernel monitor



OS has to be deprivileged.

Key requirements for intra-kernel monitor



OS has to be deprivileged.

a) Privileged instruction trapping.

b) OS memory isolation.

c) Deterministic privilege transition.

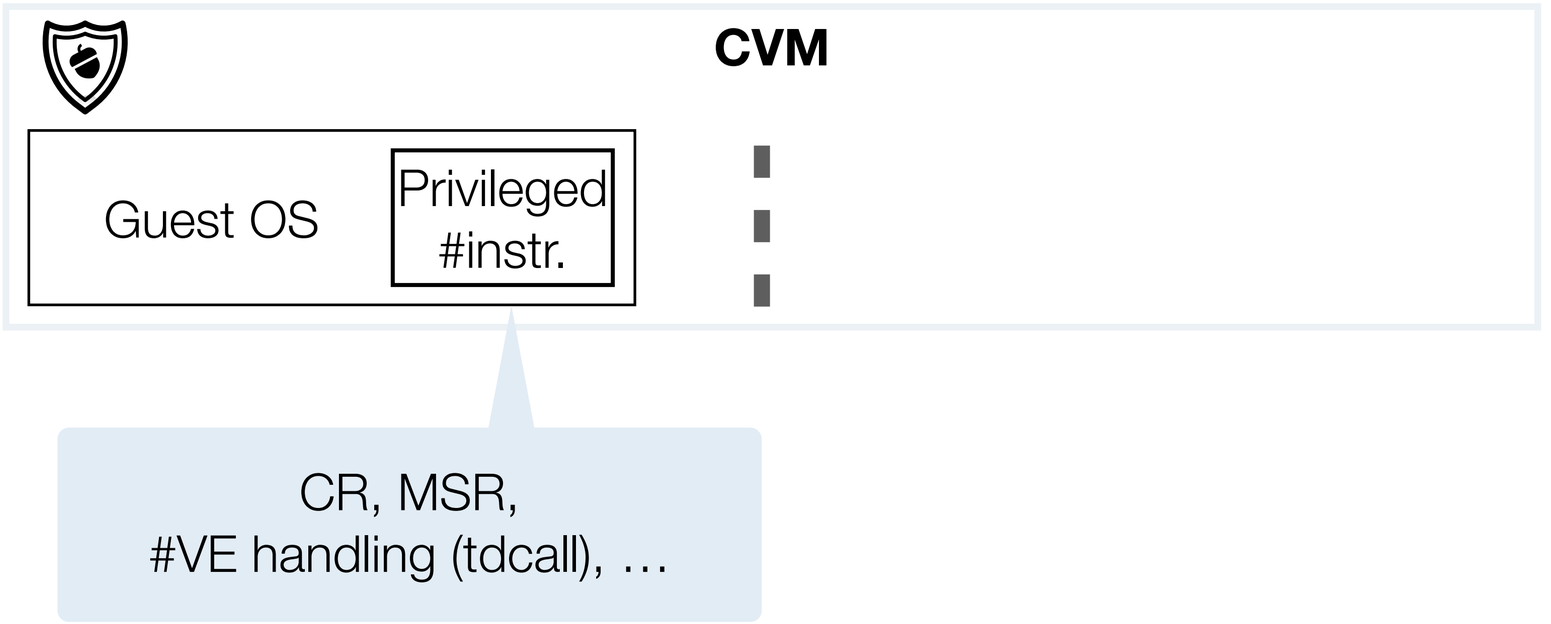
Key-enabling features:

supervisor mode execute prevention (SMEP), control-flow enforcement technology (CET).

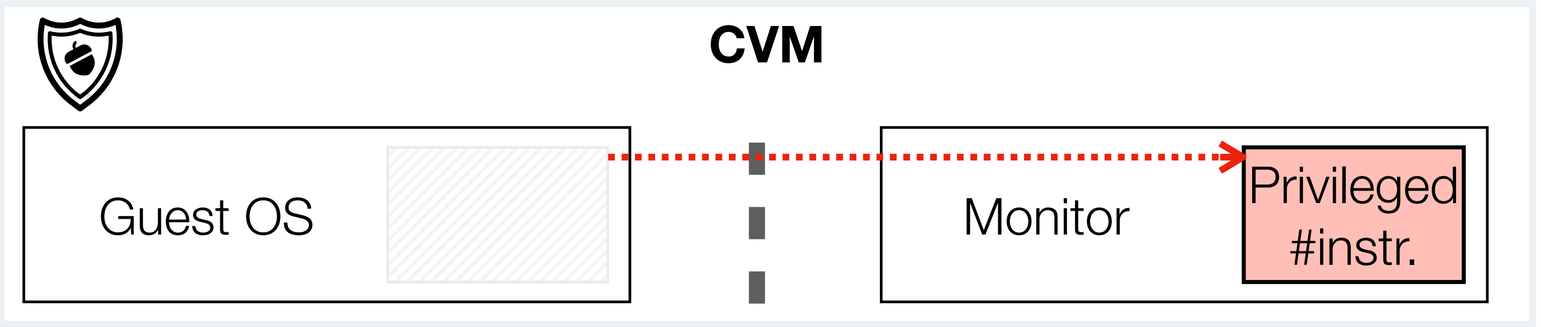
page table trapping, protection key supervisor (PKS).

Secure call gates

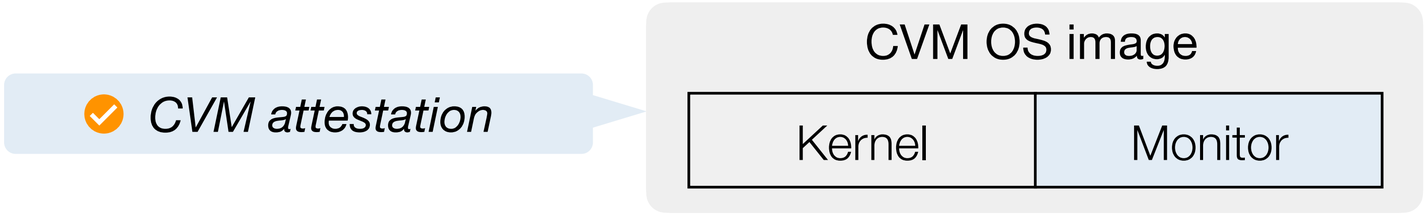
a) Trap OS’ privileged instructions



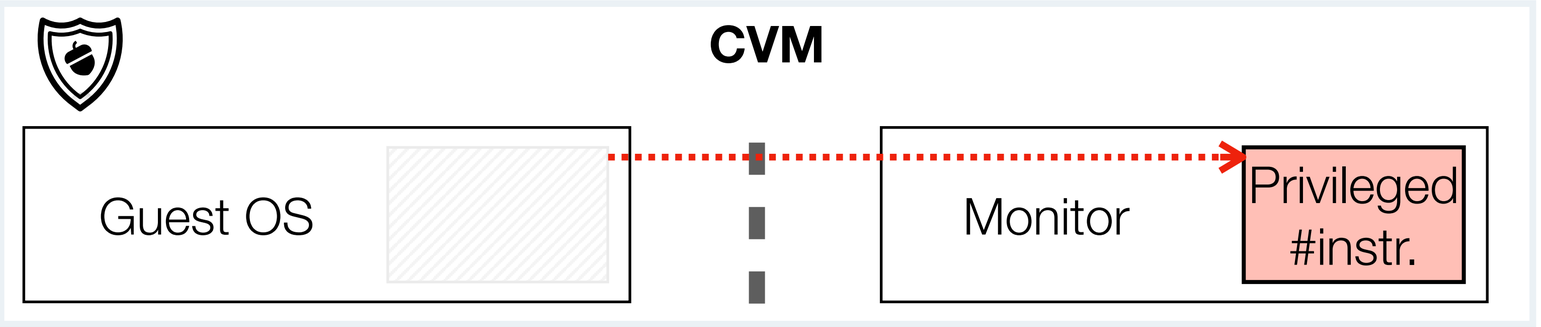
a) Trap OS’ privileged instructions

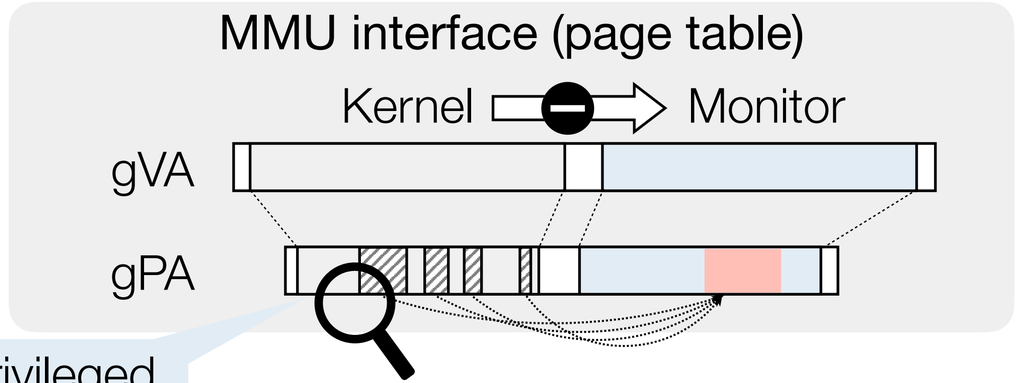


Trapped by kernel instrumentation



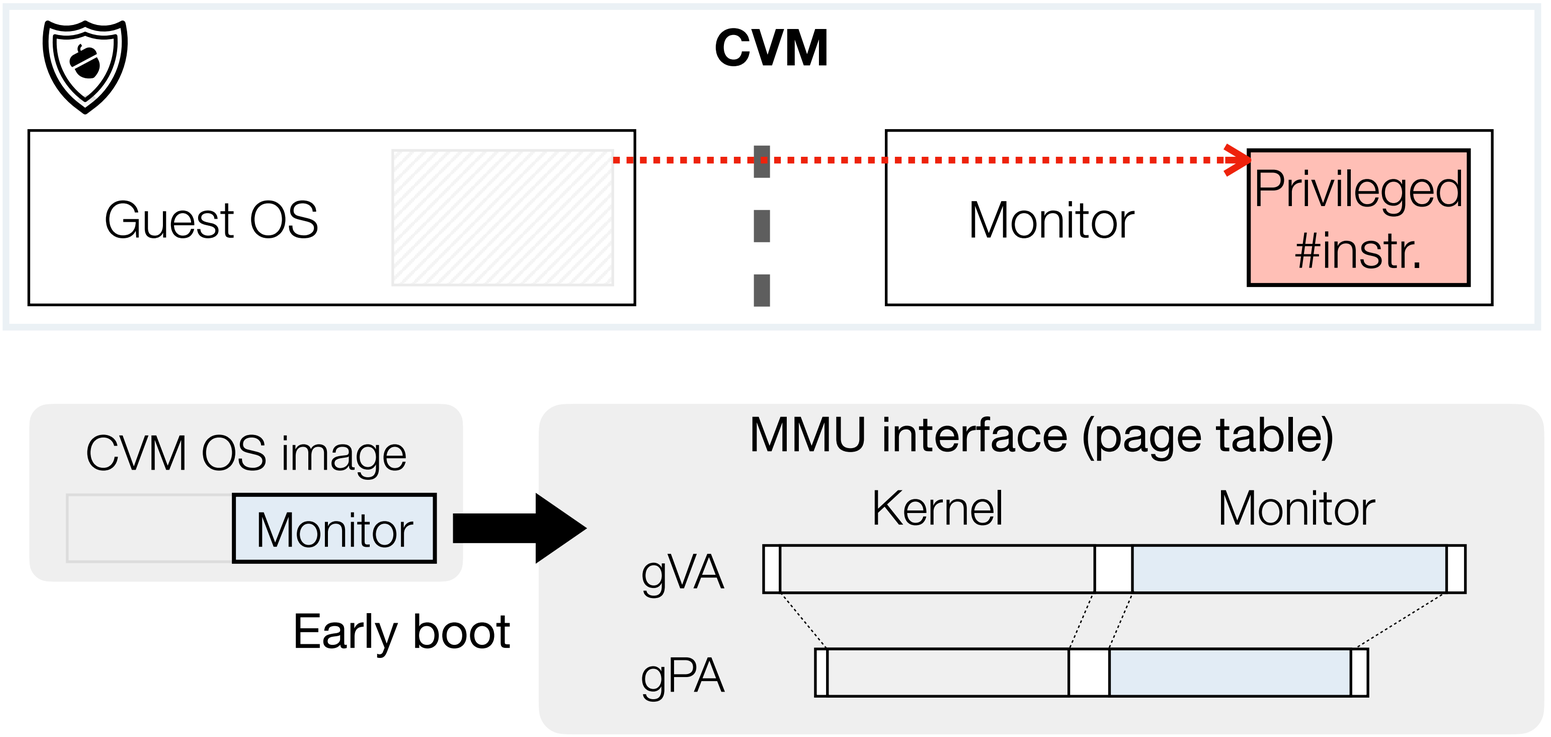
a) Trap OS' privileged instructions



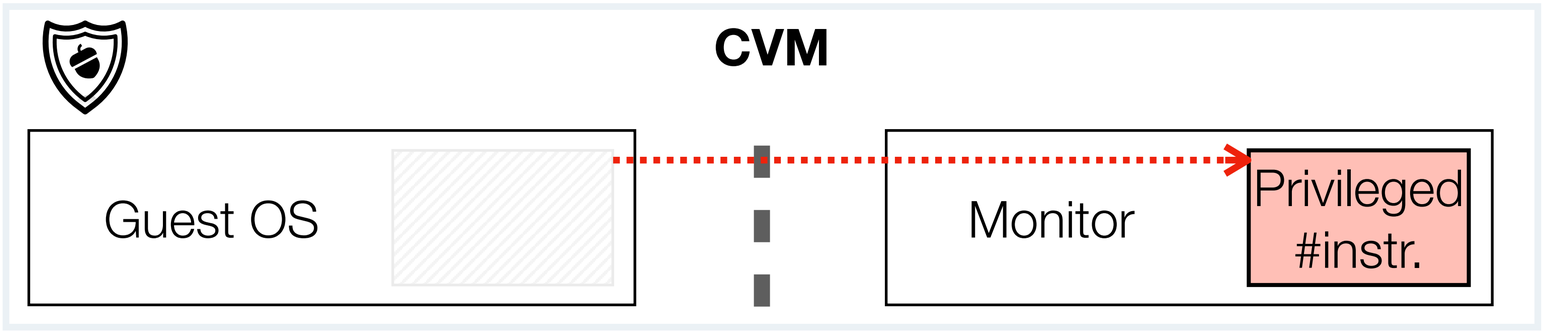


Kernel cannot execute privileged instructions
(Kernel memory W^X; SMEP)

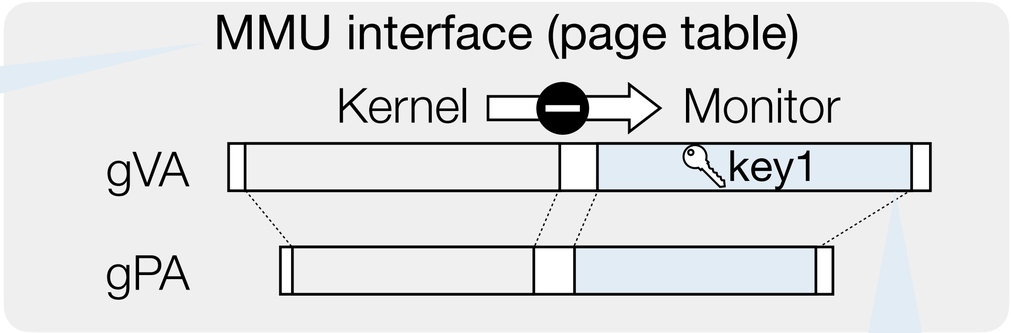
b) Isolate OS-level memory



b) Isolate OS-level memory

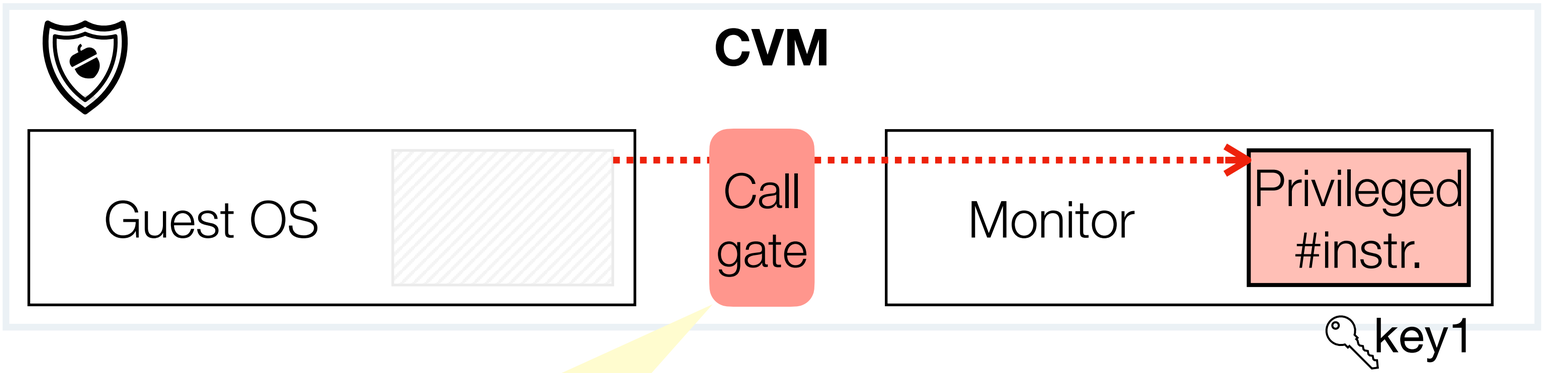


MMU (page table) updates:
trapped by the Monitor



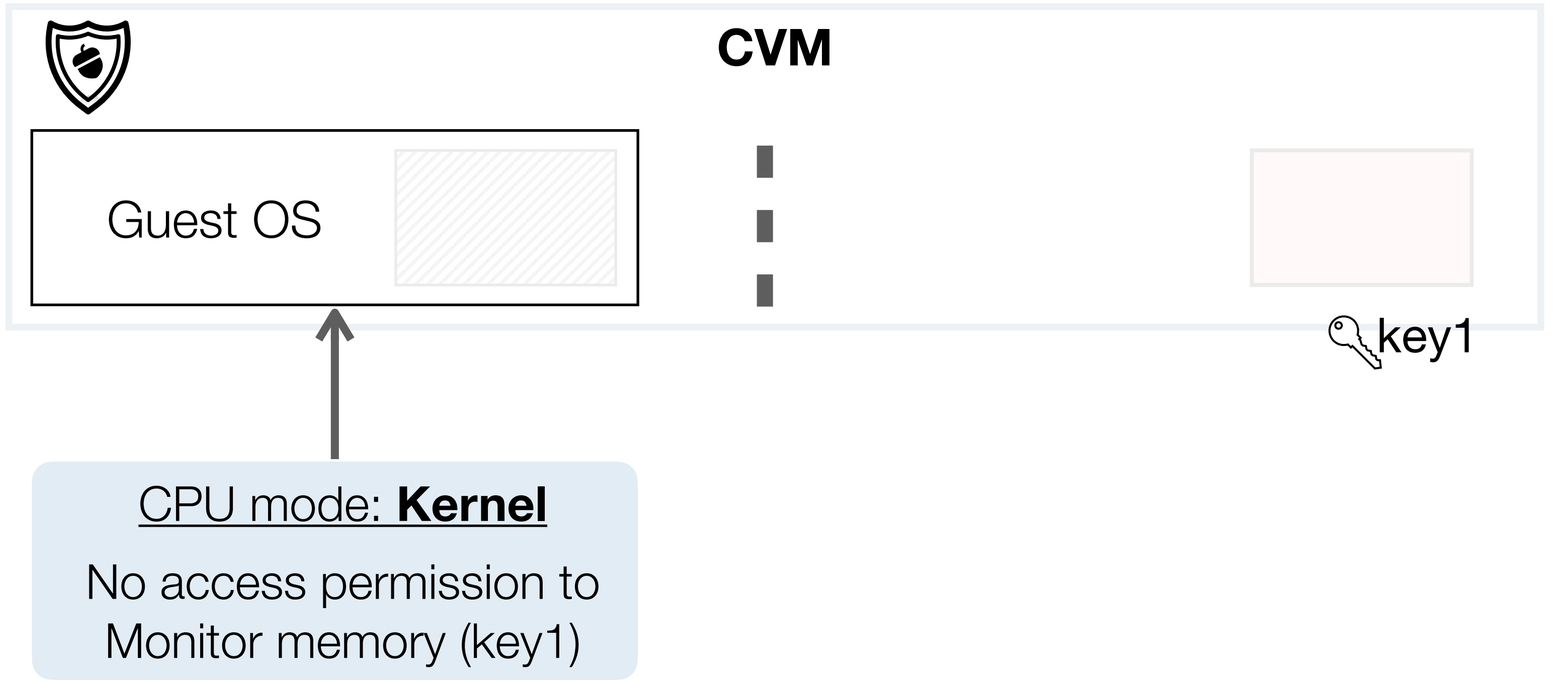
PKS: restrict OS permissions
in a per-CPU core manner.

c) Deterministic privilege transition

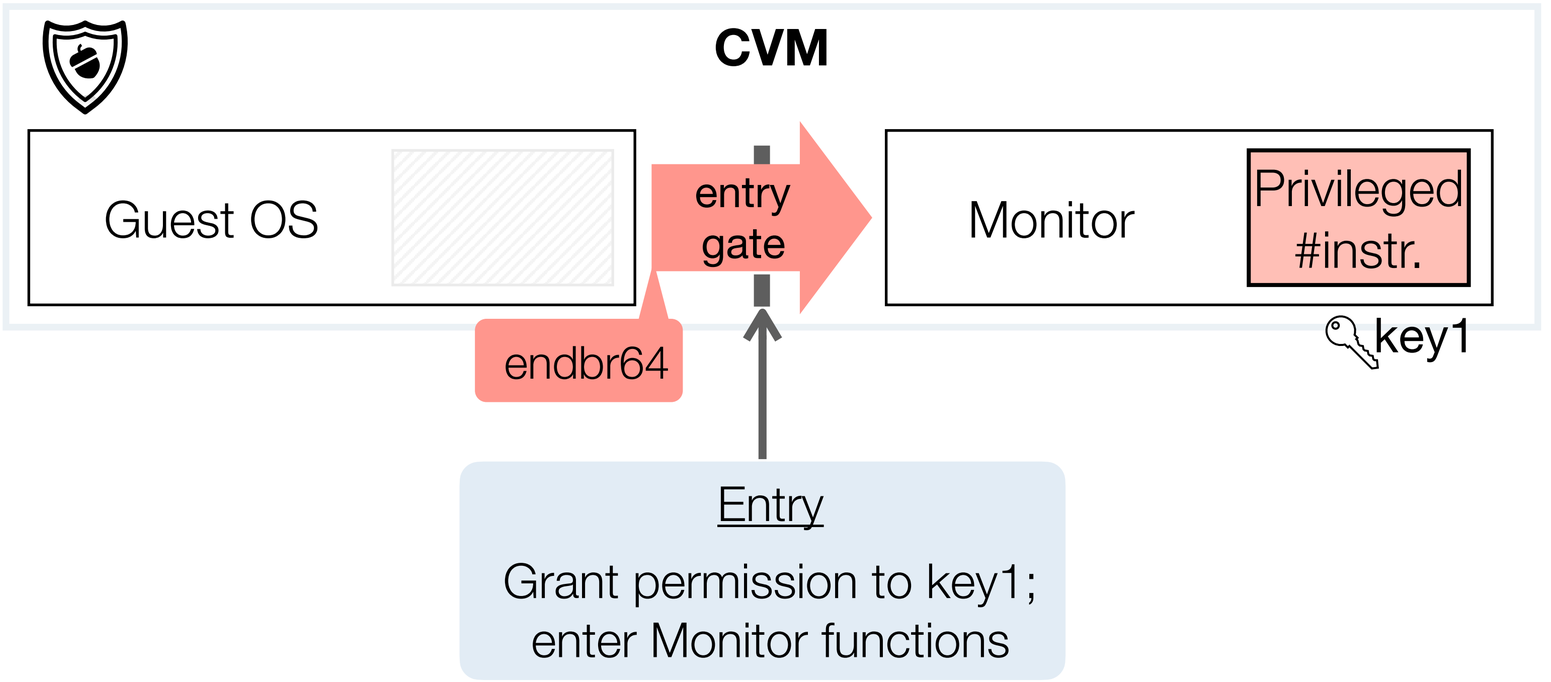


Call gate: **the only explicit** interface to request privileged instructions

c) Deterministic privilege transition

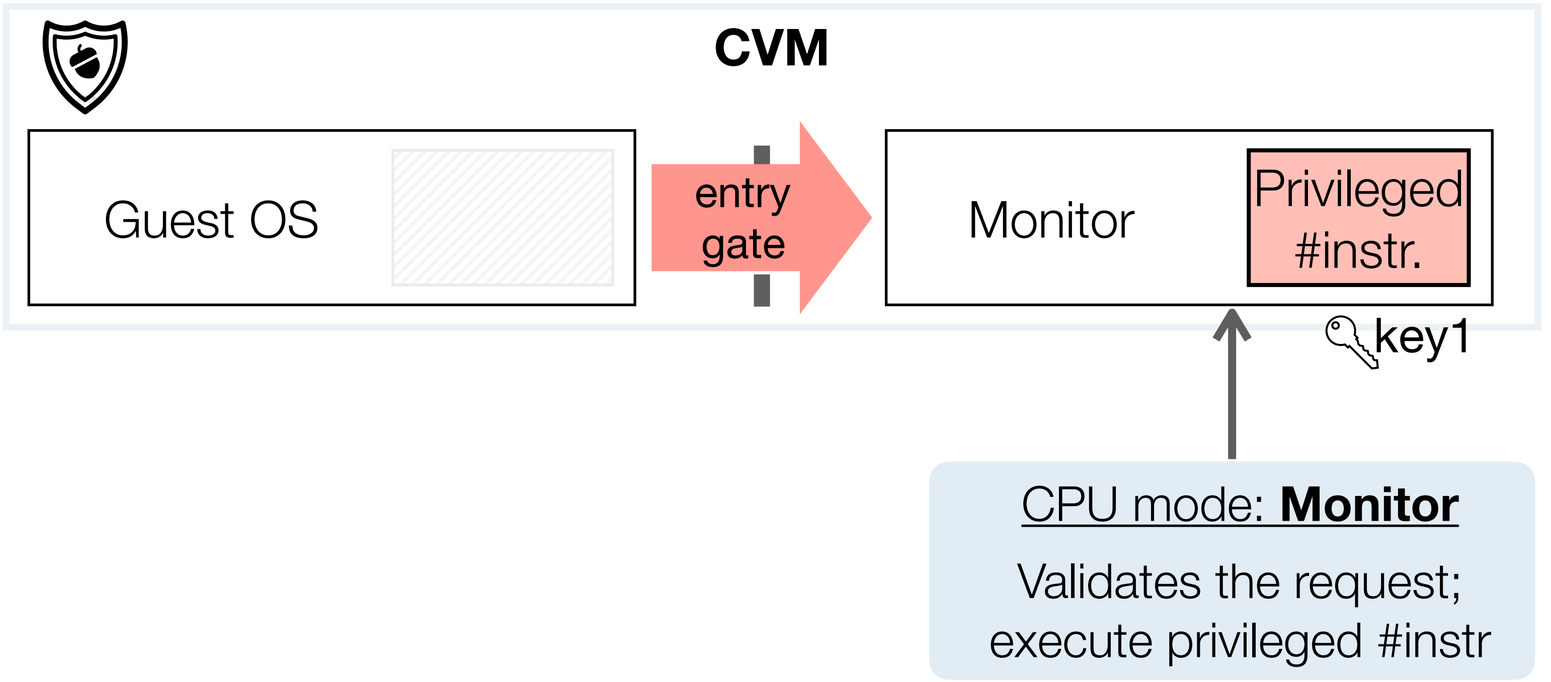


c) Deterministic privilege transition

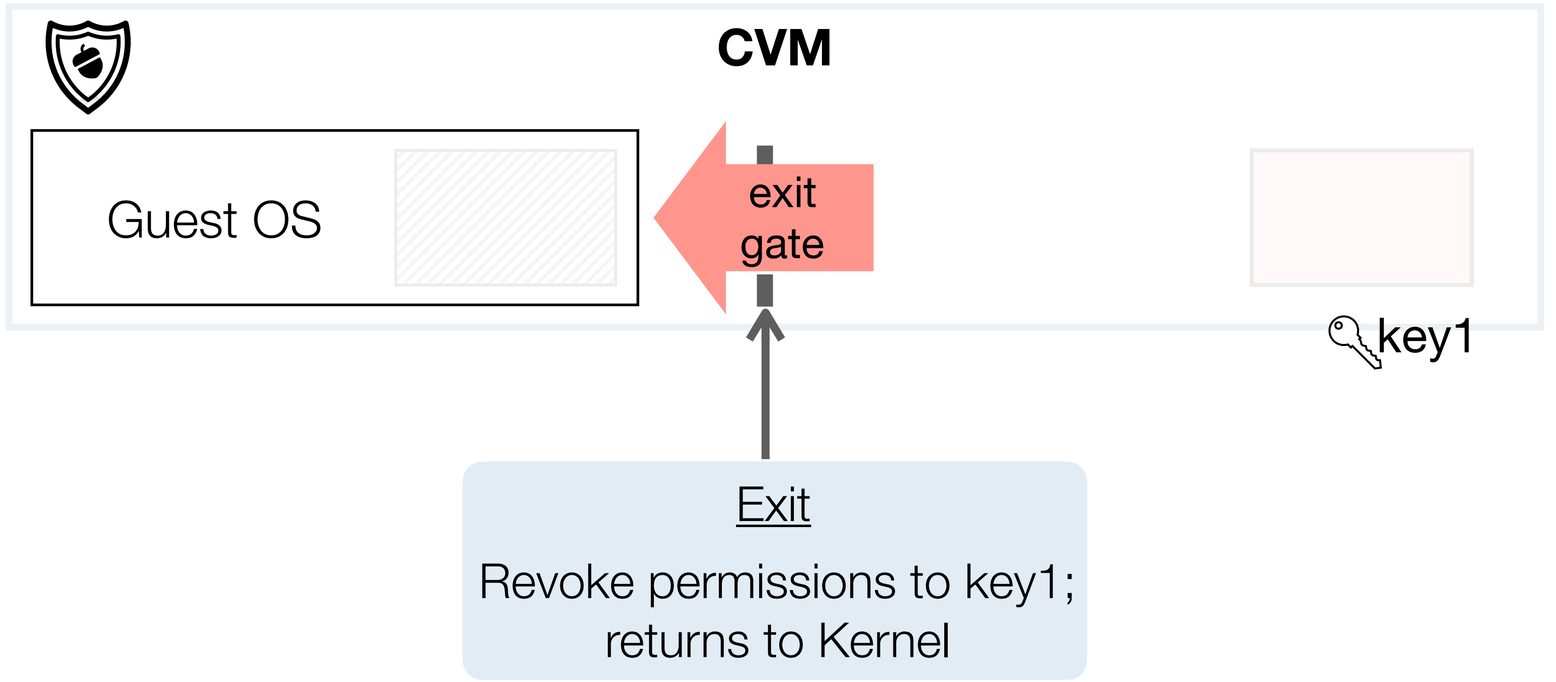


CET: the gate is **exclusive entry**

c) Deterministic privilege transition



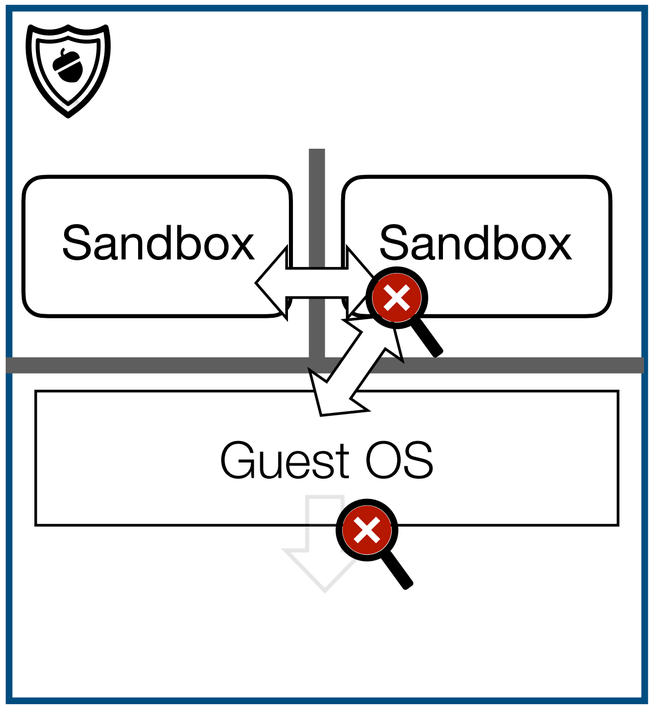
c) Deterministic privilege transition



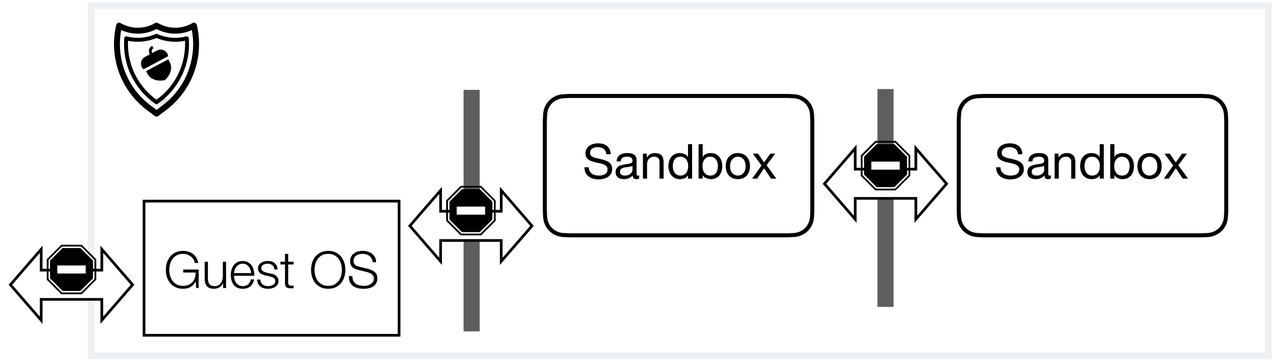
Research question 2: isolation for sandbox

Intra-CVM isolation

How to confine sandbox execution?



Key requirements to confine execution for sandbox



Sandbox and outside interactions have to be confined.

a. Memory channel isolation.

b. Runtime OS service support.

c. Software exit channel isolation.

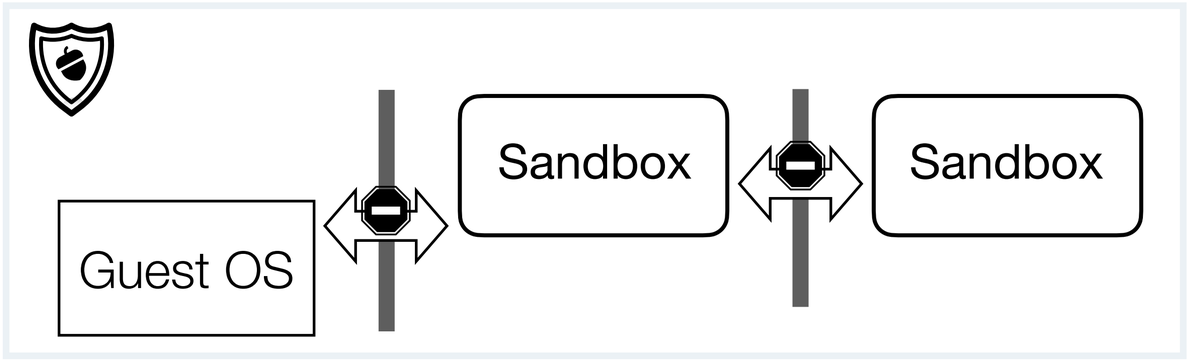
Key-enabling features:

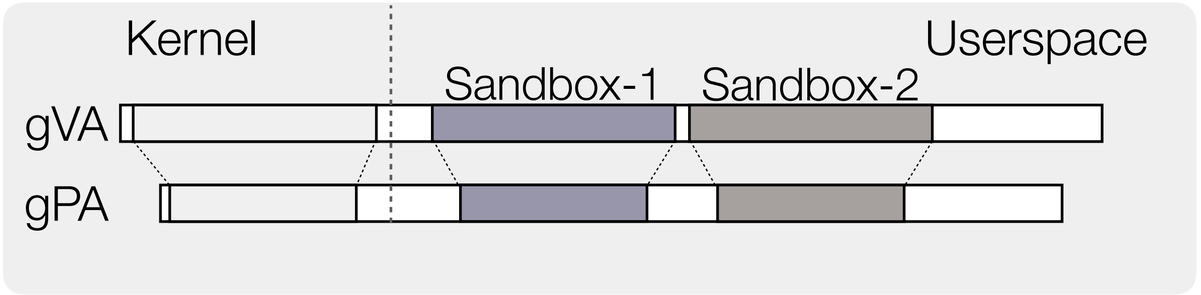
page table trapping,
supervisor mode access prevention (SMAP).

LibOS

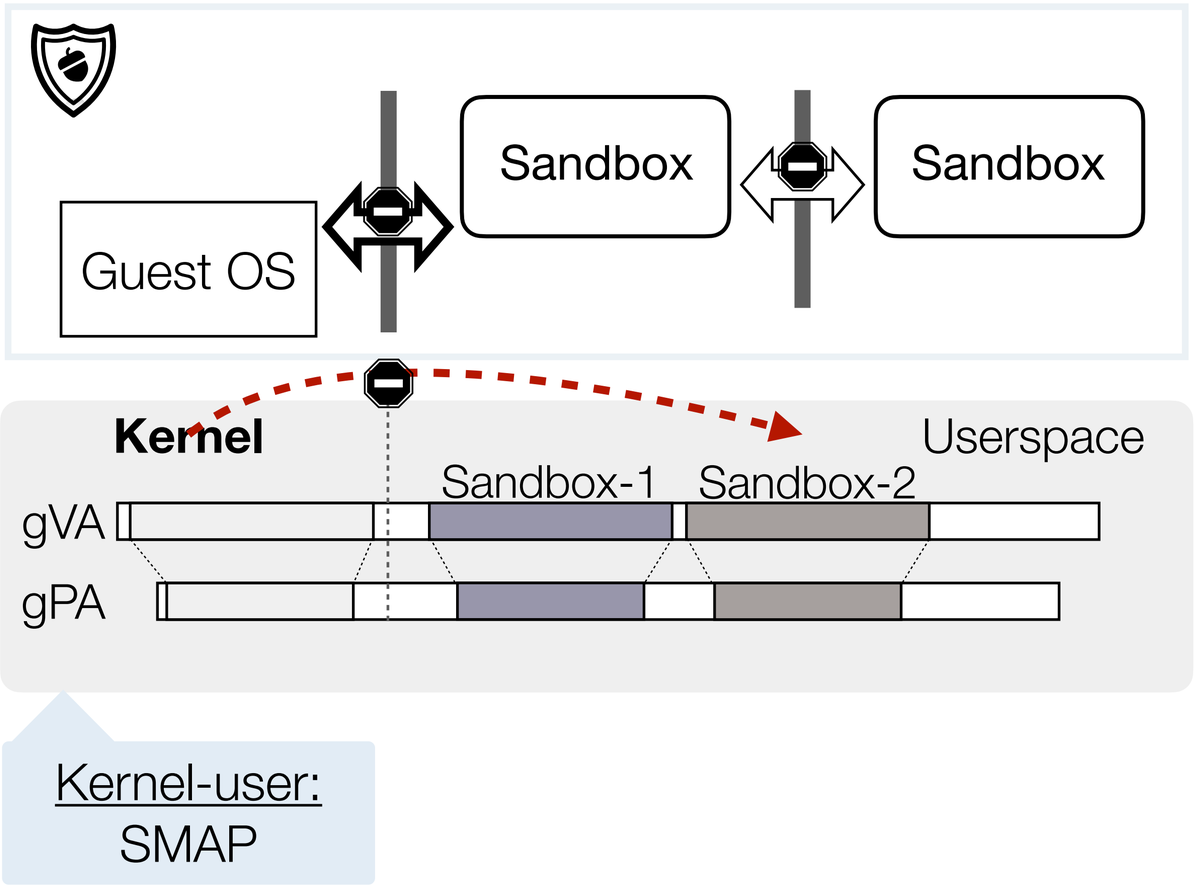
context switch interposition.

a) Isolate memory channels of sandbox



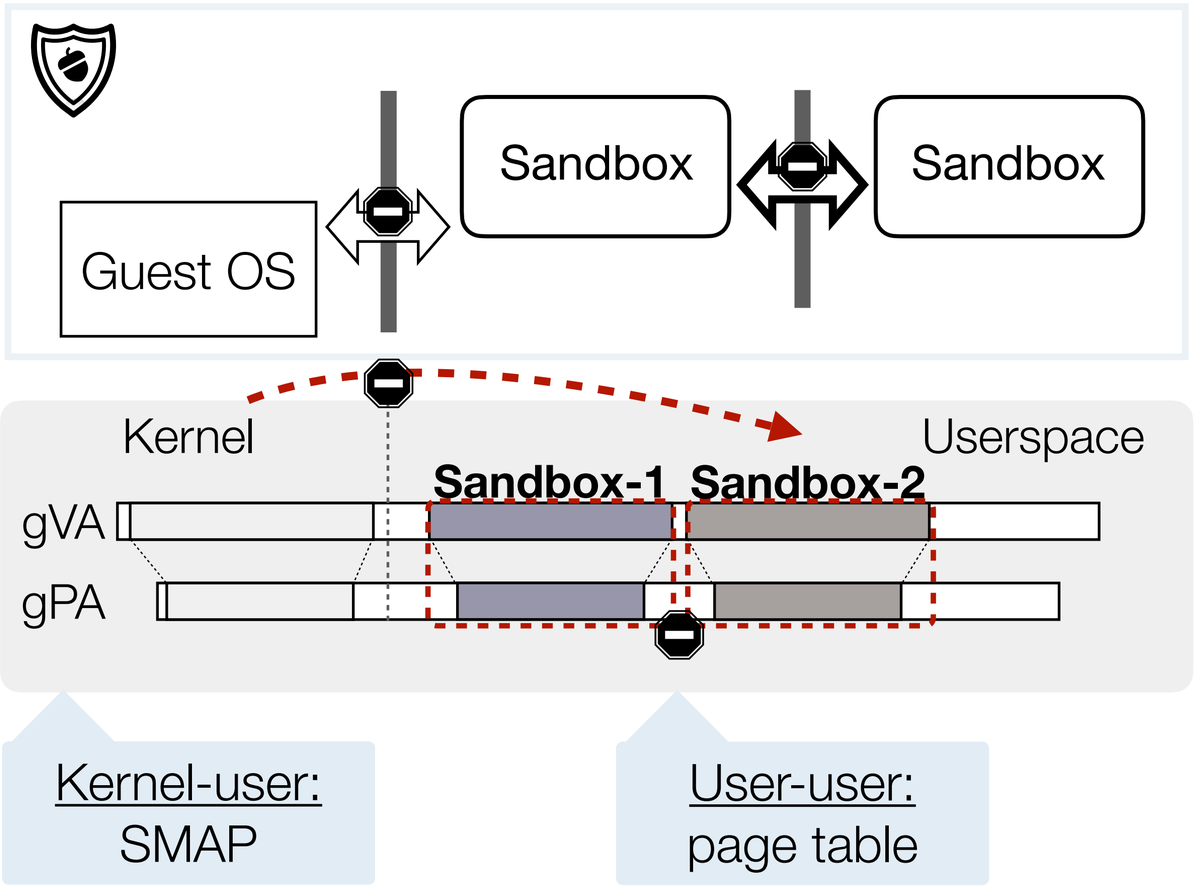


a) Isolate memory channels of sandbox



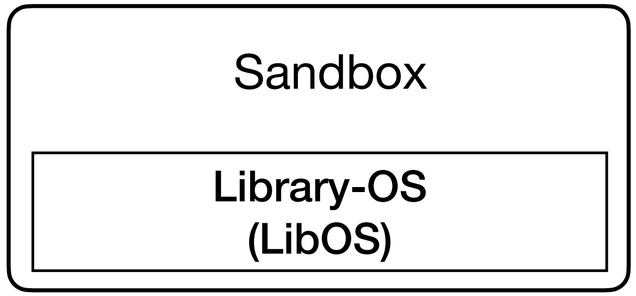
Kernel-user:
SMAP

a) Isolate memory channels of sandbox



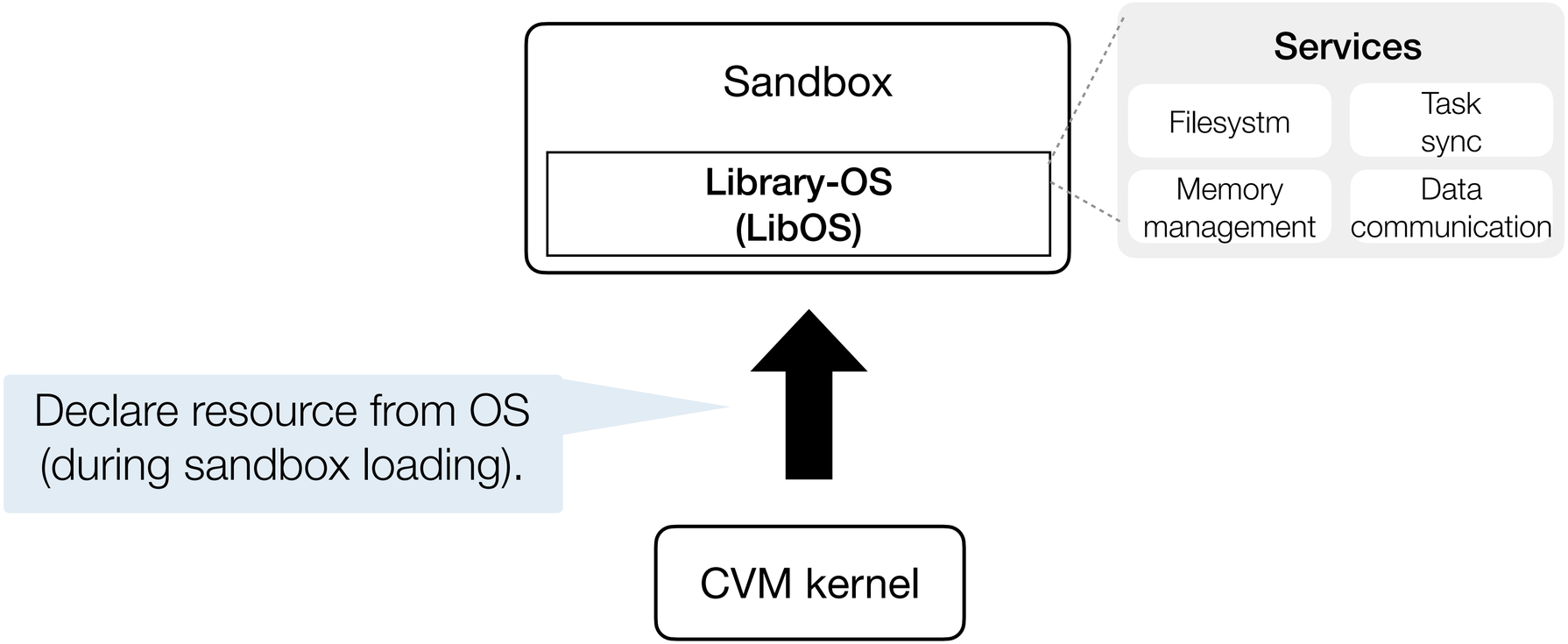
b) Emulate runtime system services in LibOS

Required OS services are sandbox self-contained.



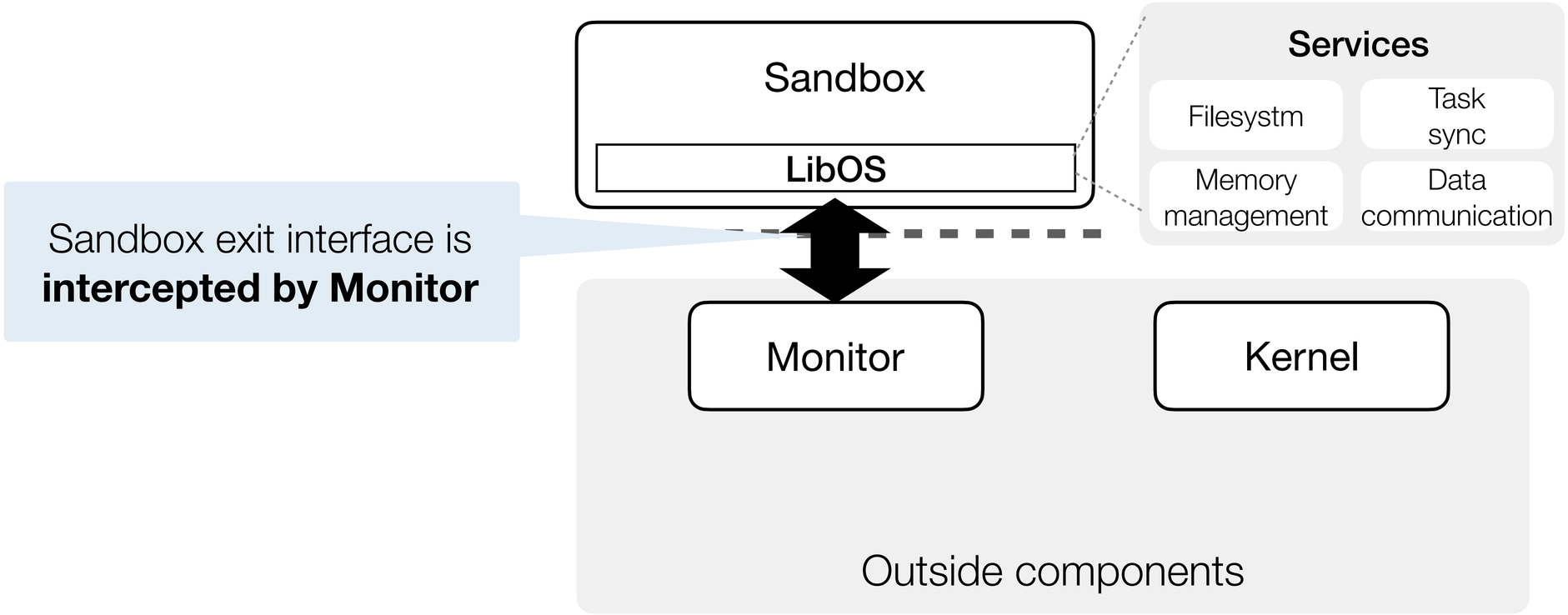
b) Emulate runtime system services in LibOS

Required OS services are sandbox self-contained.



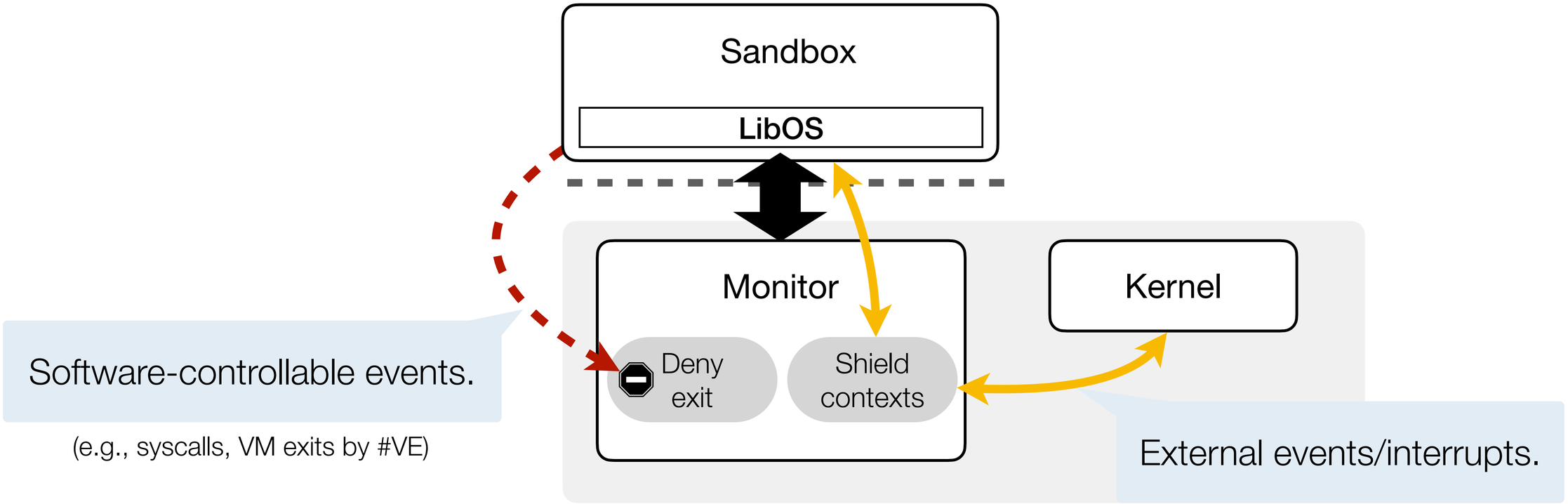
c) Restrict sandbox exit channel

Avoid exiting to the outside to disclose data.

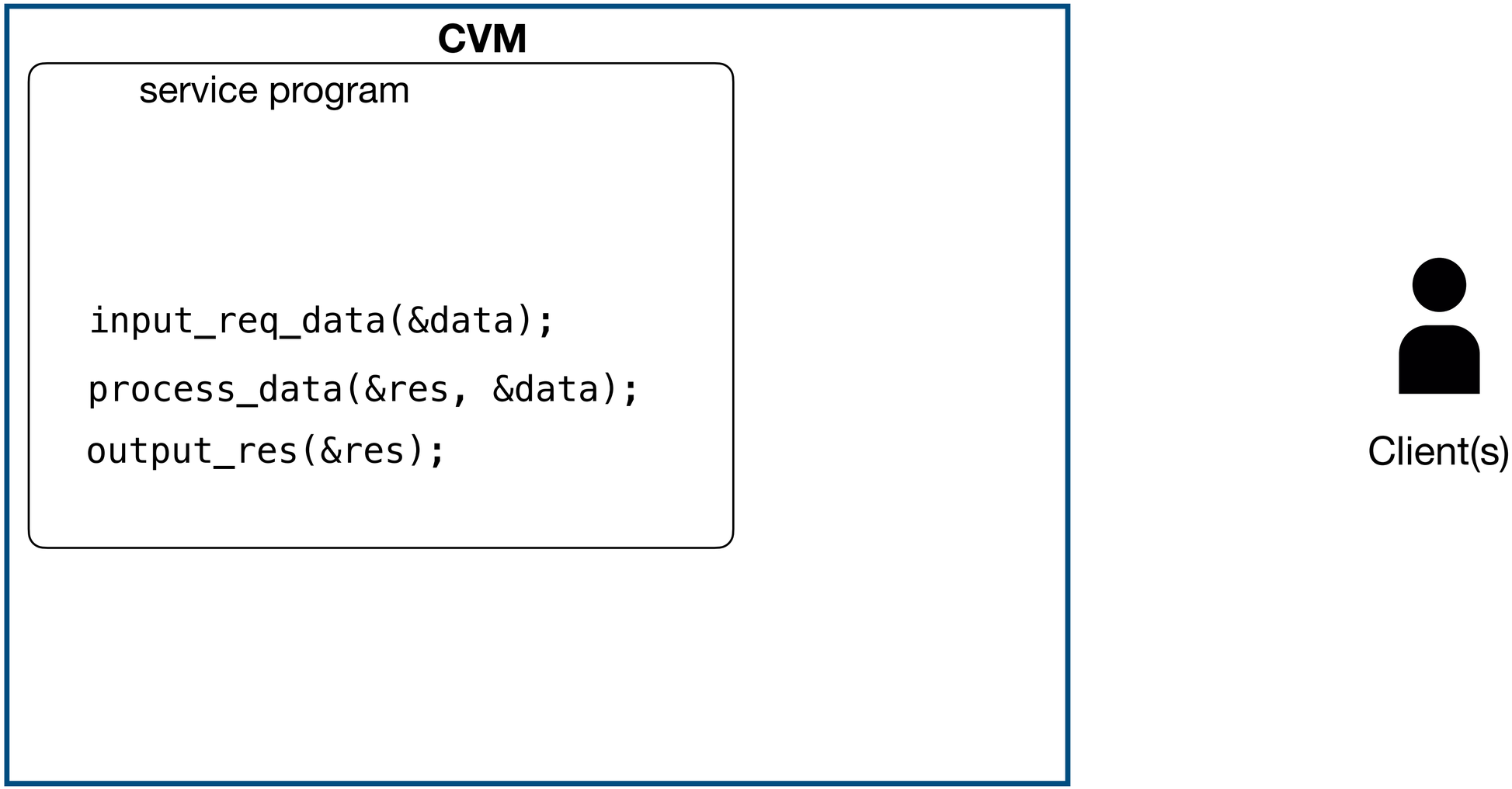


c) Restrict sandbox exit channel

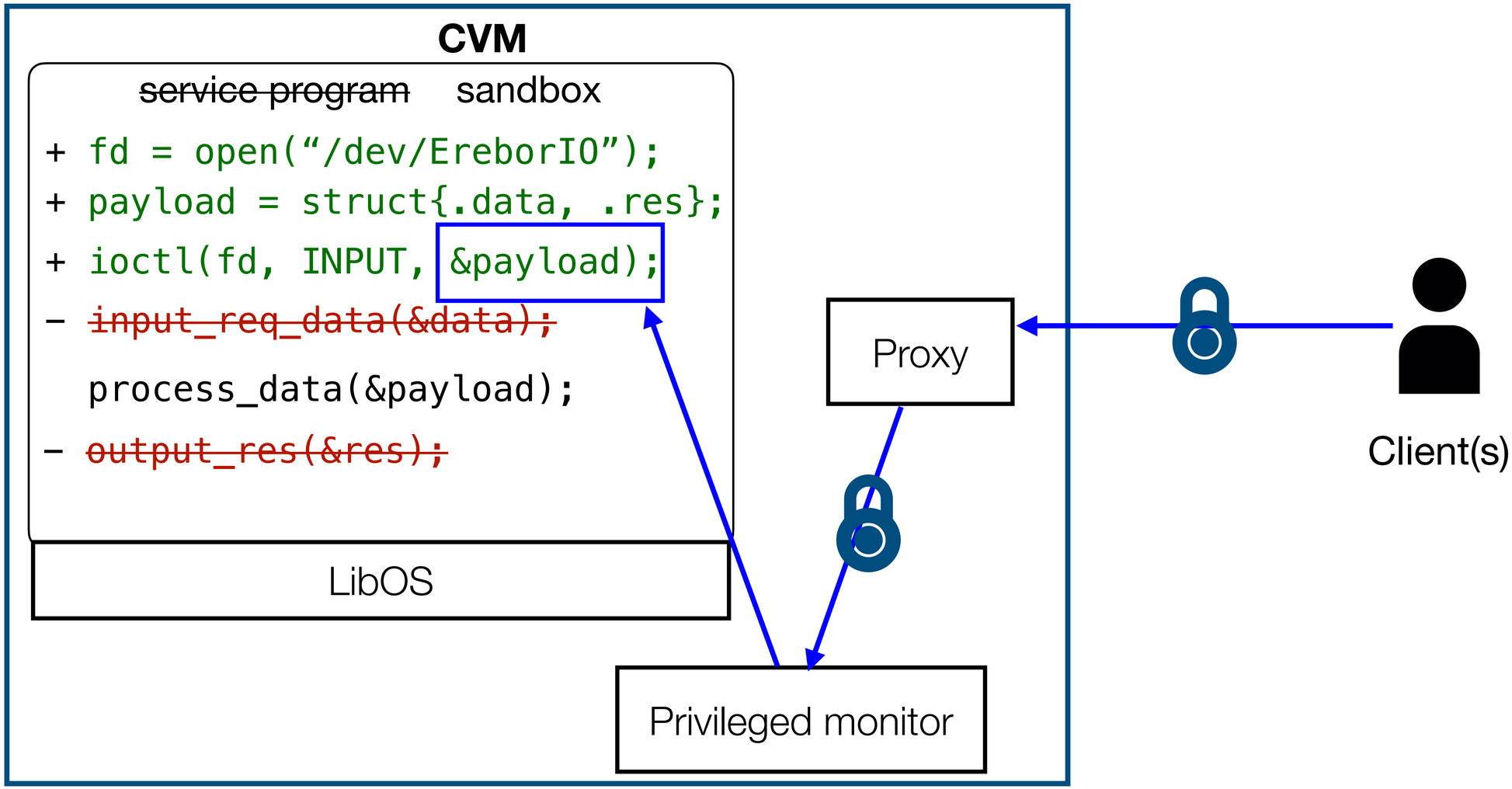
Avoid exiting to the outside to disclose data.



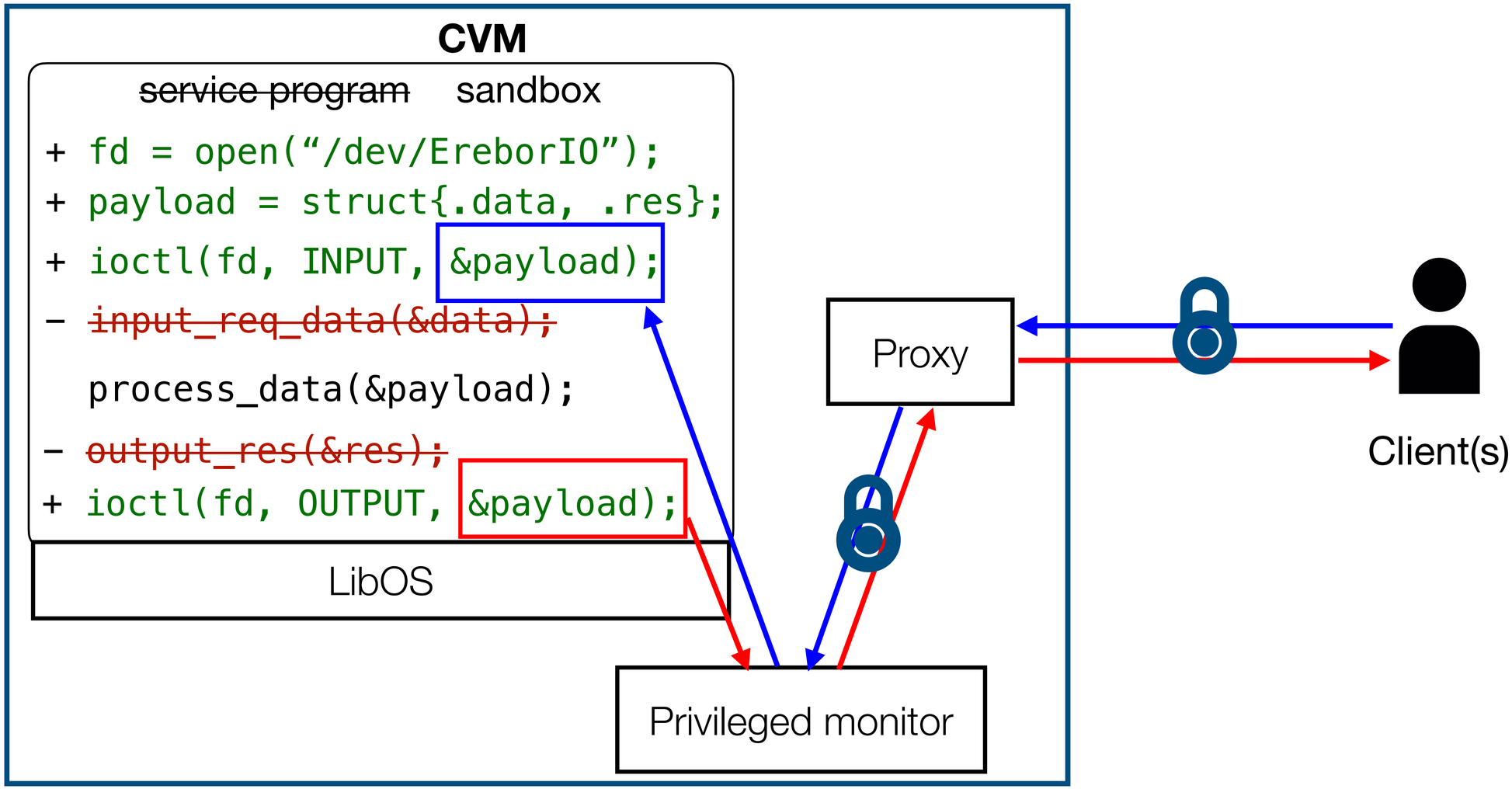
Port a service program to a sandbox container



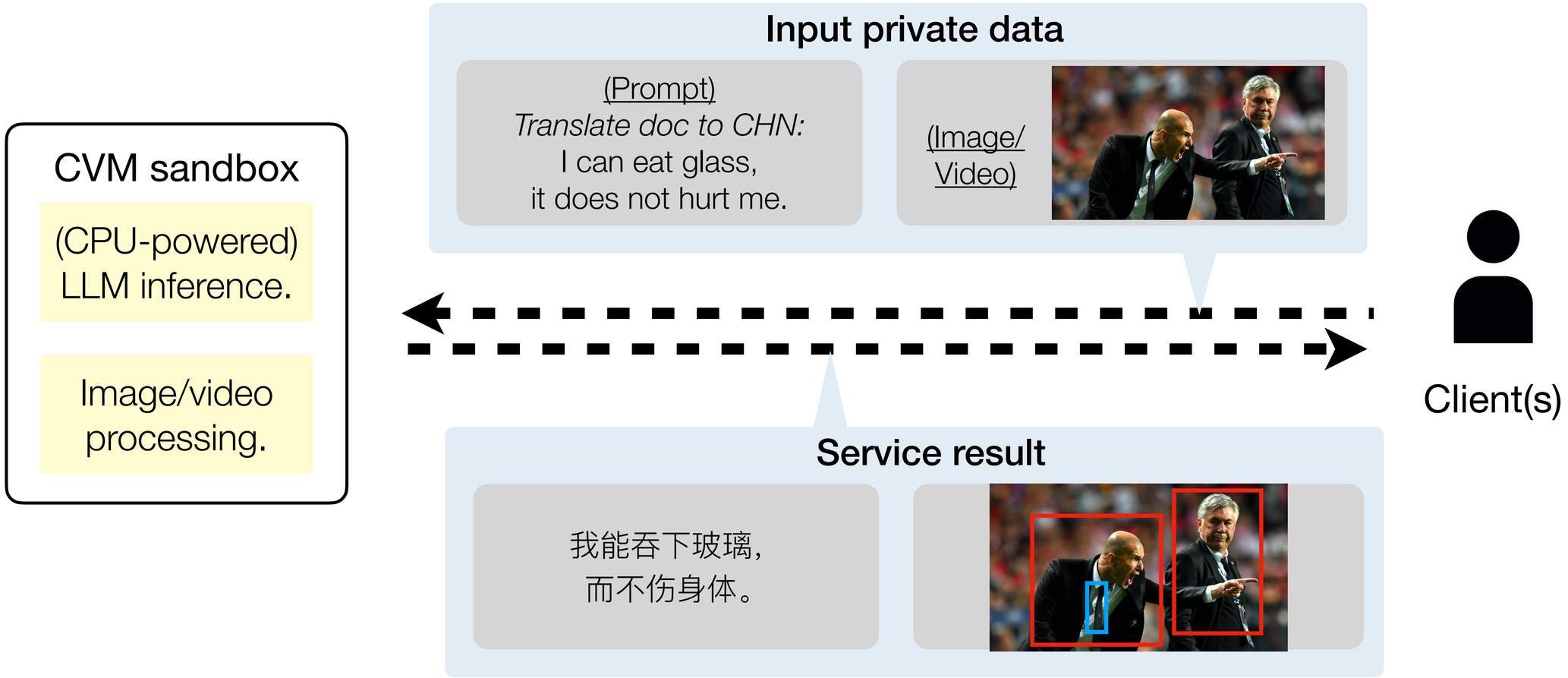
Port a service program to a sandbox container



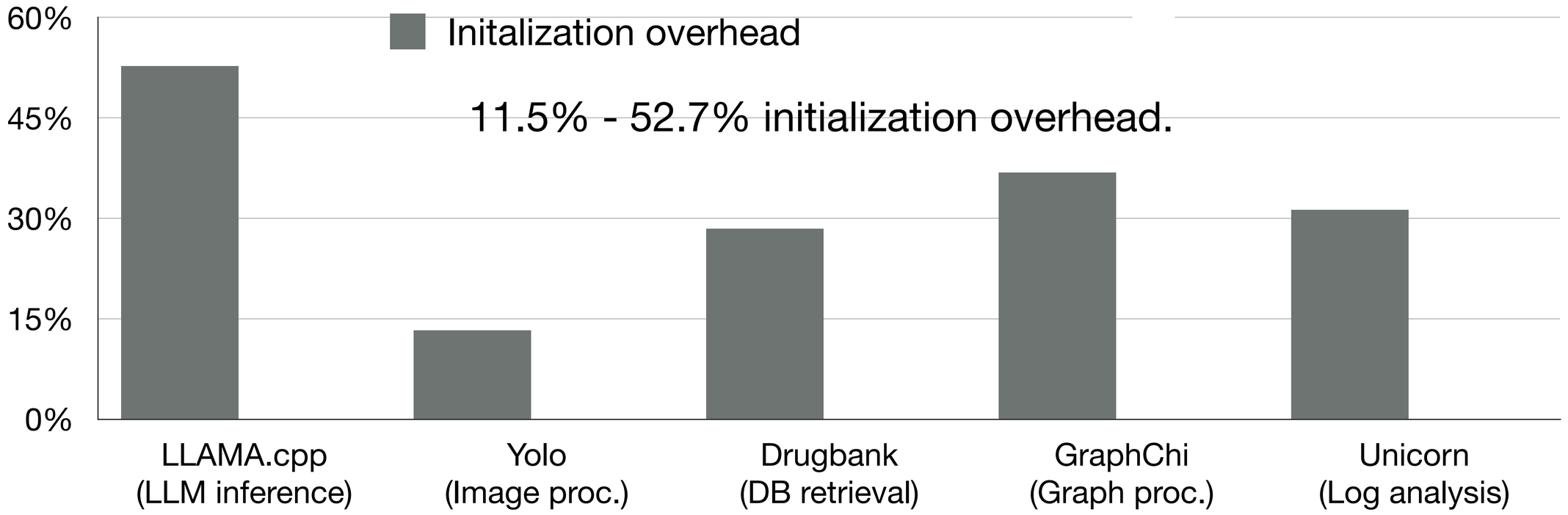
Port a service program to a sandbox container



Example sandbox service scenarios



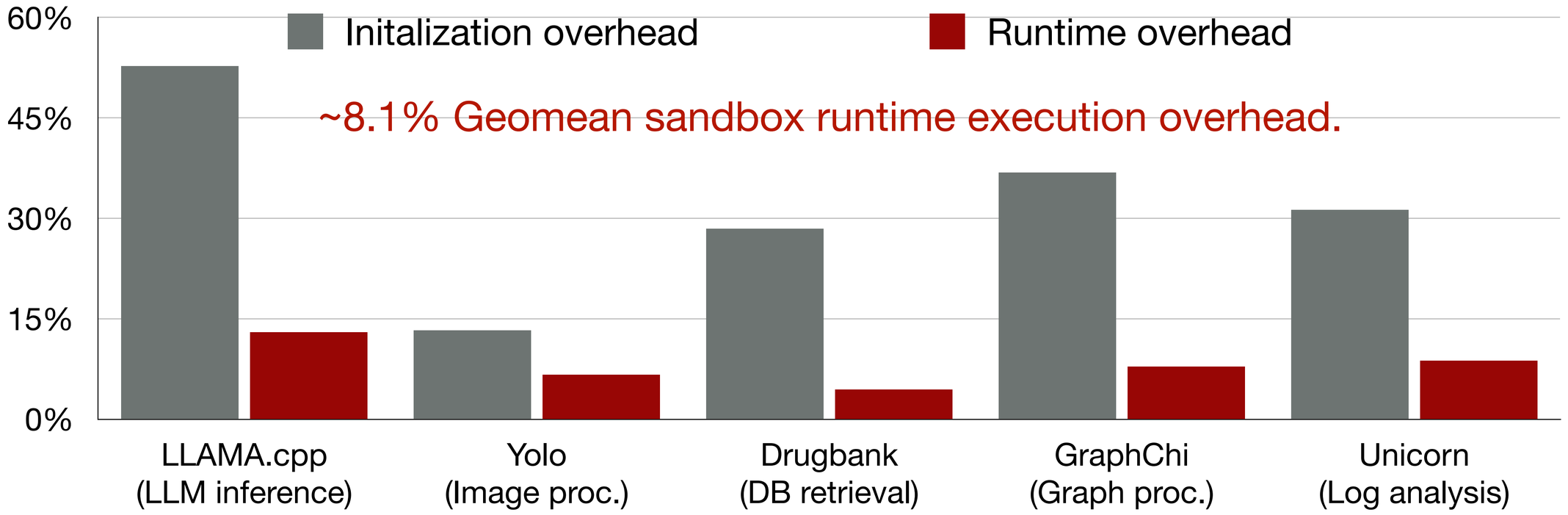
Sandbox application overhead



Cause: **privileged instruction traps**
(primarily MMU updates)
during memory initialization.

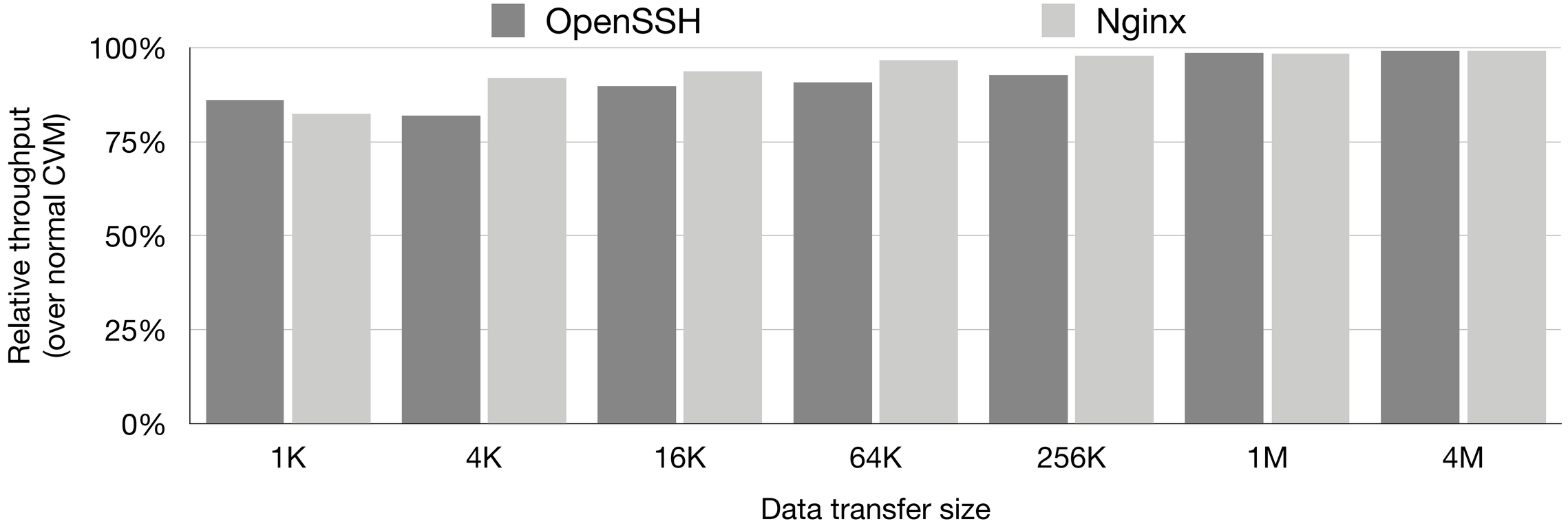
This is a one-time effort
that can be optimized **by warm-start**.

Sandbox application overhead



Low runtime cost:
resource pre-declaration & LibOS-emulation.

CVM normal program overhead

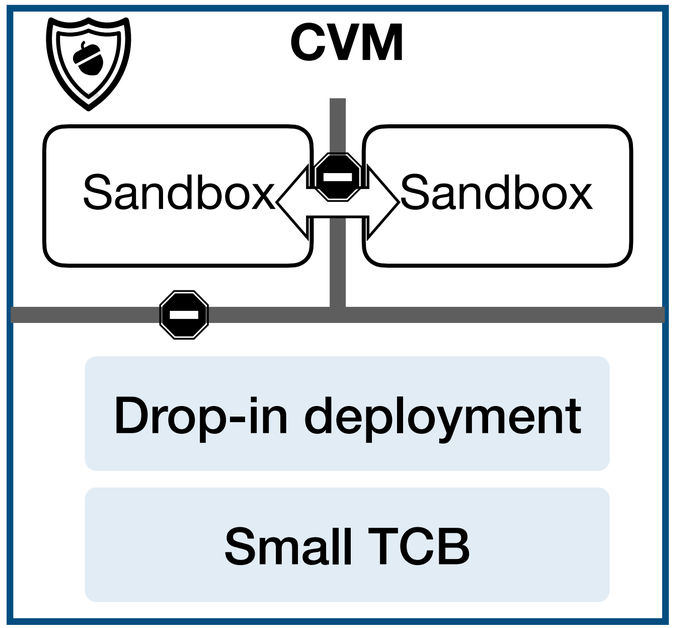


On average 8.2% (OpenSSH) and 5.1% (Nginx) throughput downgrading (under intensive workloads).

Takeaways

Confidential VM: No client data privacy guarantee in third-party-owned SaaS platforms.

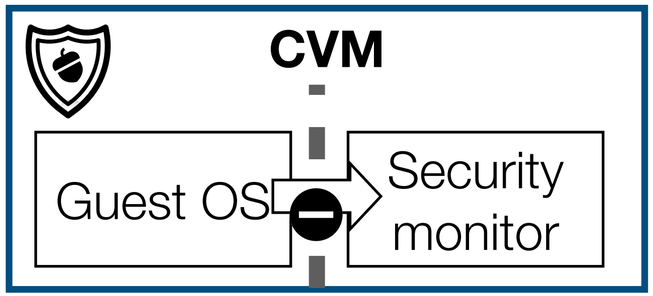
Erebor:



Code at

Questions?

Backup: hardware primitives



Protection Key Supervisor (PKS)

Per-core, kernel page-level memory permission (RW) control

Each page table entry has a *key (0 - 15)*

Key	Frame
0	PFN_0
1	PFN_1
…	
15	PFN_i

CPU's model-specific register controls its permission to each key's memory

IA32_PKRS *vCPU-1*

IA32_PKRS *vCPU-0*

Key	15	…	1	0
Perm	-/-	…	R/-	R/W

Control-flow Enforcement Technology (CET)

Per-core, both forward and backword control flow enforcement