CSIE 5310

Virtualization Security

Prof. Shih-Wei Li

Department of Computer Science and Information Engineering
National Taiwan University

Deprivileging Hypervisors: HypSec [1]

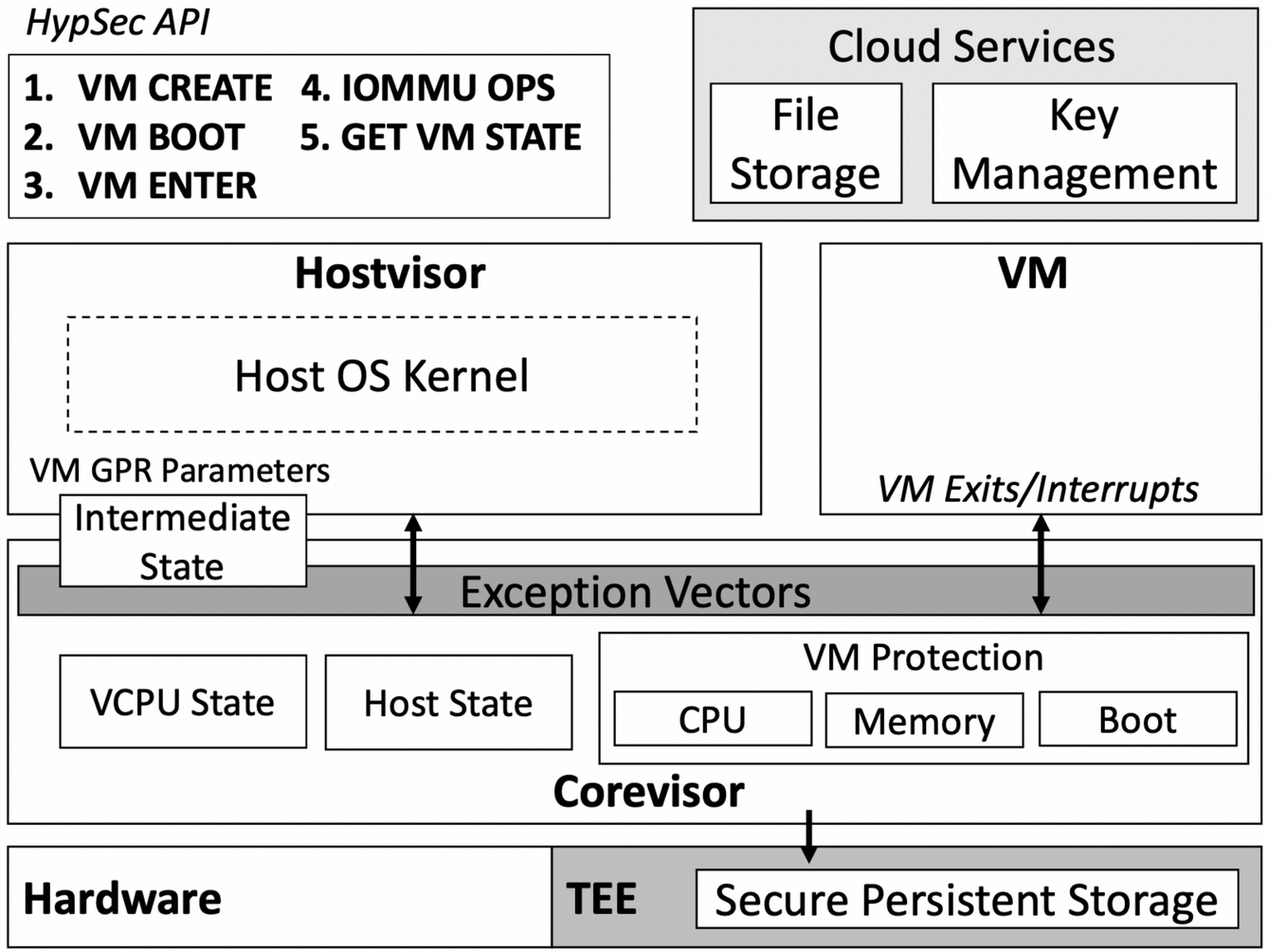
- **Goal:** protect VM confidentiality and integrity against a malicious hypervisor
- **Threat model:** assume attackers fully control the hypervisor; assume attackers cannot physically steal the machine
- **Mechanism and Policy:** employ microkernel approach to retrofit an exiting hypervisor; applies HypSec to retrofit KVM
  - Interposes VM exits/entries to protect VM
  - Employs an access control approach to protect VM data

[1] Protecting Cloud Virtual Machines from Hypervisor and Host Operating System Exploits, Shih-Wei Li, John S. Koh, Jason Nieh, USENIX Security 19

HypSec: Observation & Design Intuition

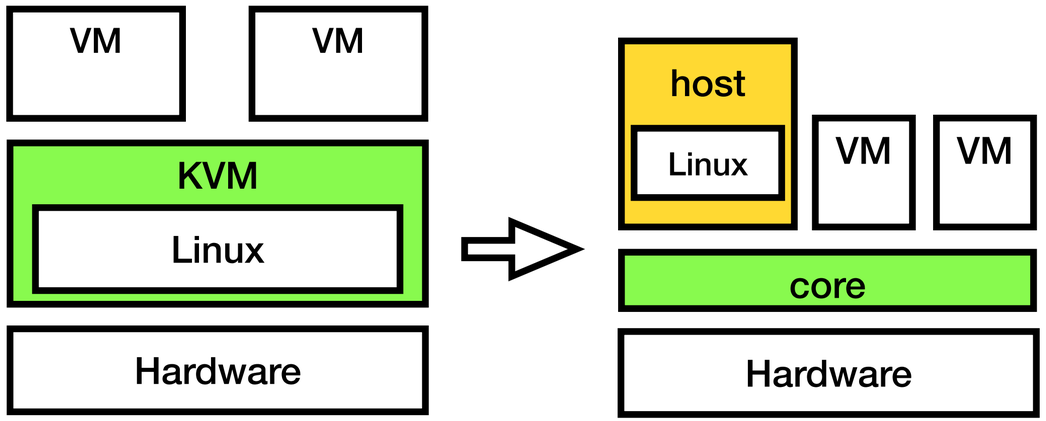
- Observation:
  - Applications increasingly employ an end-to-end approach to protect their I/O data
  - Many hypervisor functions do not need access to VM data
    - Ex: VCPU scheduling and memory allocation
- Design Intuition:
  - Separate access control from resource allocation
  - Focus on CPU and memory protection; rely on VMs to protect I/Os

Deprivileging Hypervisors: HypSec Design

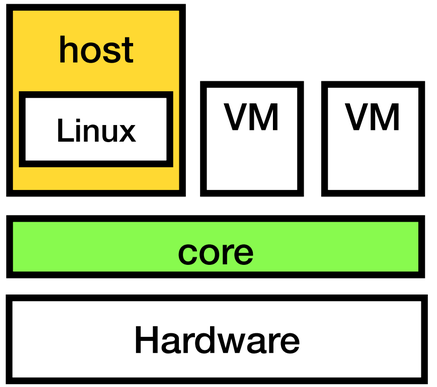


HypSec: KVM Retrofit

- Retrofit the monolithic KVM hypervisor into with two parts:
  - Small **corevisor** that protects VM data
  - Large set of ***untrusted*** services **(hostvisor)** that includes Linux to provide complex functionalities
- KVM implementation leverages Arm VE



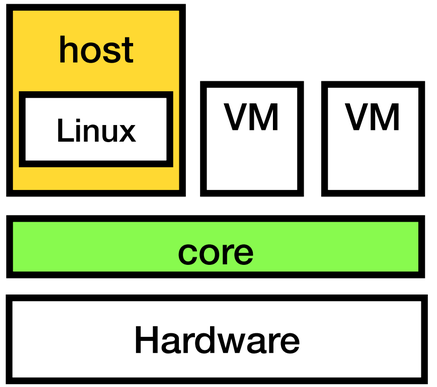
HypSec: KVM Retrofit



- Corevisor has full hardware access
  - Protects VM data in CPU registers and memory
- Hostvisor has restricted hardware access
  - Provides functionality that requires no access to unencrypted VM data
- Relies on VMs protect their I/O data
  - Employ encrypted networking (TLS/SSL) or virtual disk

HypSec: KVM Retrofit

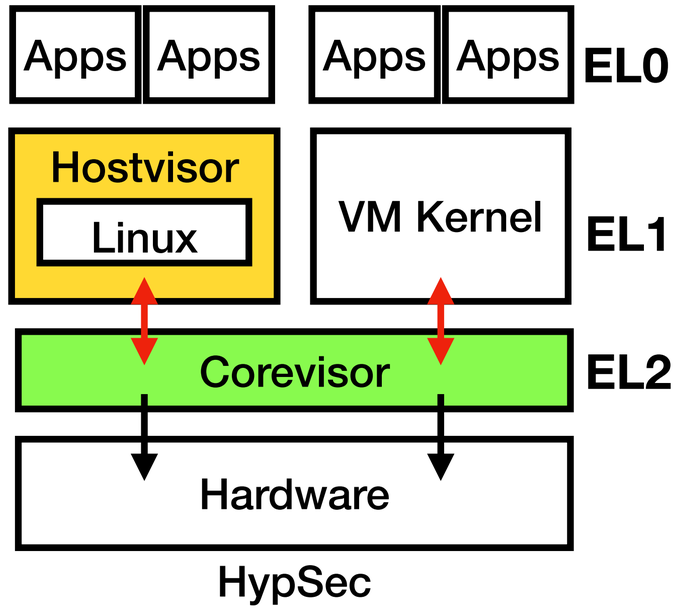
- Corevisor provides
  - CPU virtualization, page table management
  - Access control for VM data in CPU and memory
- Hostvisor provides
  - I/O virtualization
  - Resource allocation and scheduling



HypSec: Leveraging Hardware Virtualization Extensions

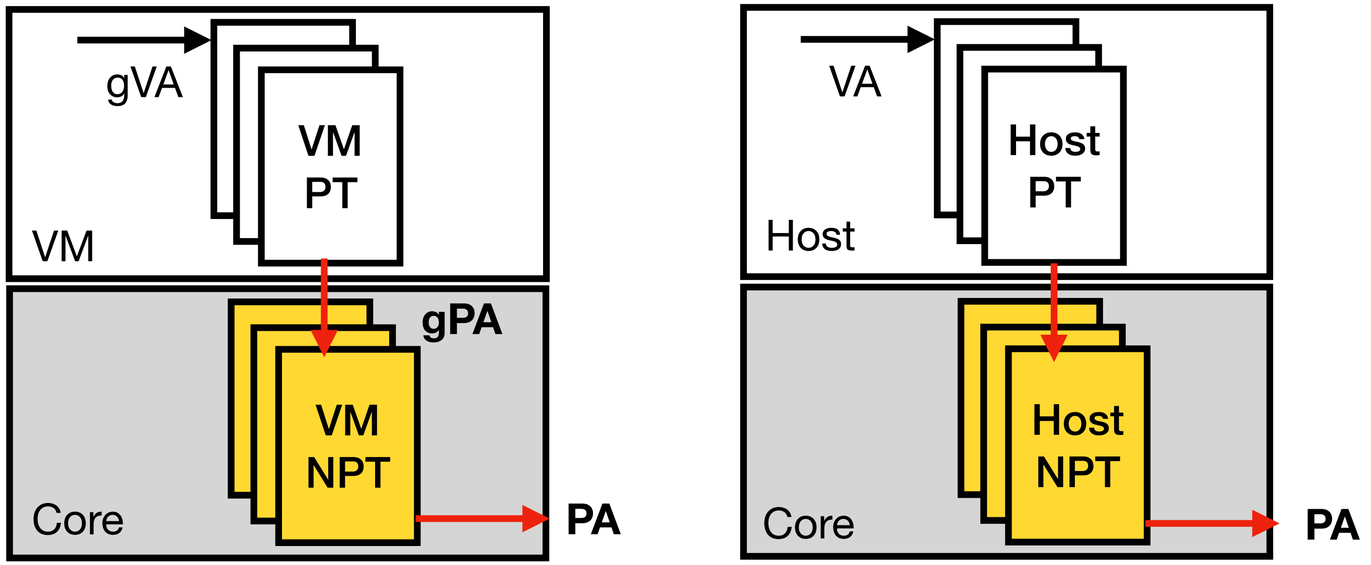
- HypSec implementation leverages Arm virtualization extensions to:
  - Deprivilege hostvisor in EL1
  - Run corevisor in EL2 to protect VM data
  - Control hardware features for VM protection

ARM®
Virtualization Extensions (VE)



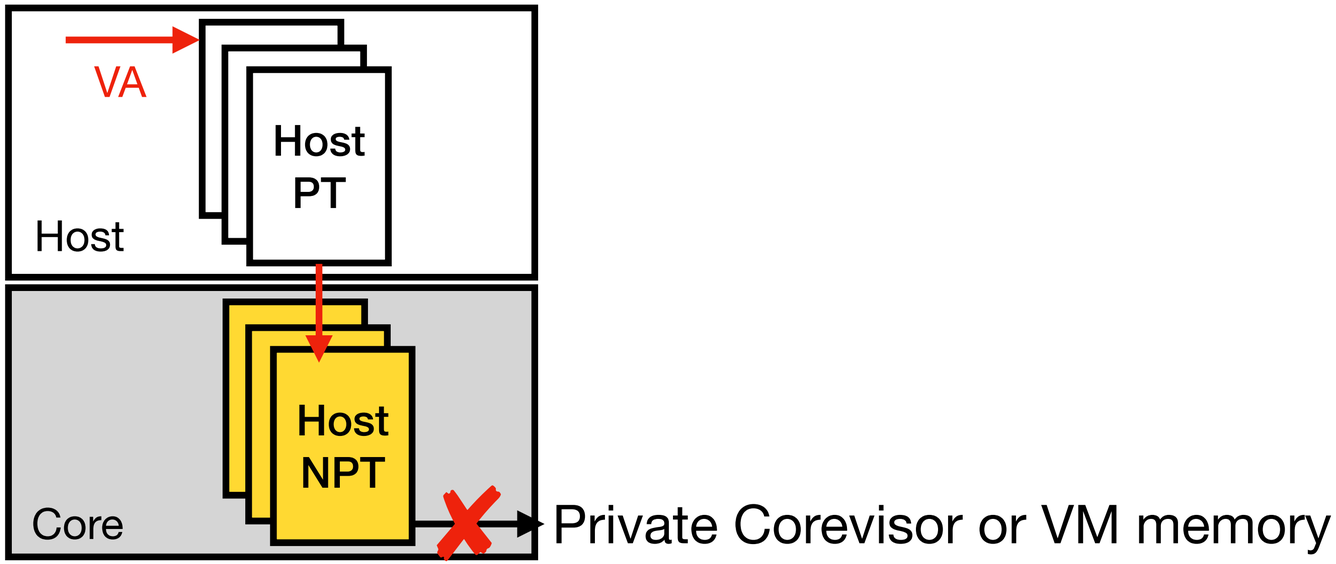
Memory Protection

- Corevisor leverages Nested Page Tables (**NPTs**)
  - Use the nested level translation to enforce memory access control



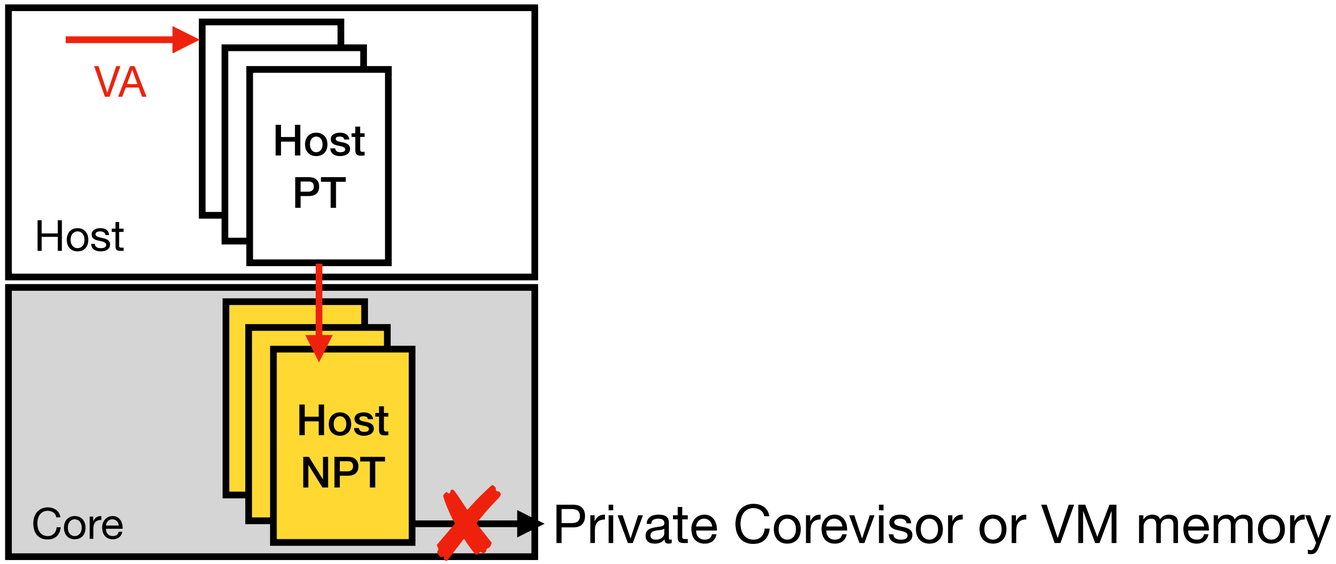
Memory Protection

- Corevisor manages NPT for Hostvisor to control the host's memory accesses
  - Ensure the host NPT does not map to corevisor's or VM's private memory



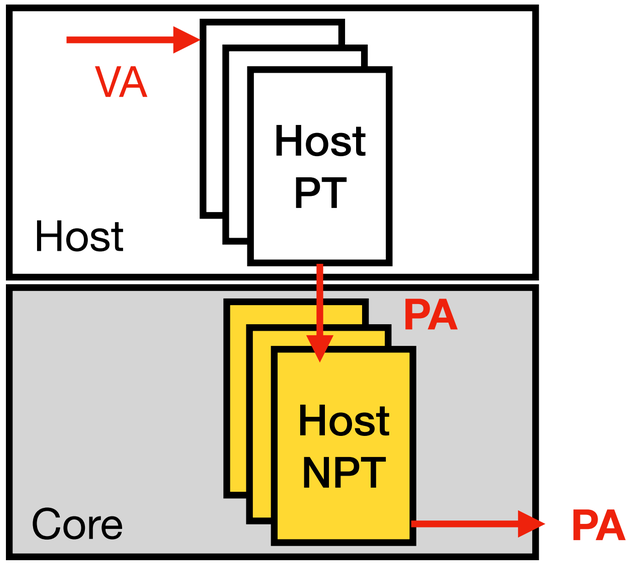
Memory Protection

- Corevisor manages NPT for Hostvisor to control the host's memory accesses
  - Ensure the host NPT does not map to corevisor's or VM's private memory
  - Any issues?



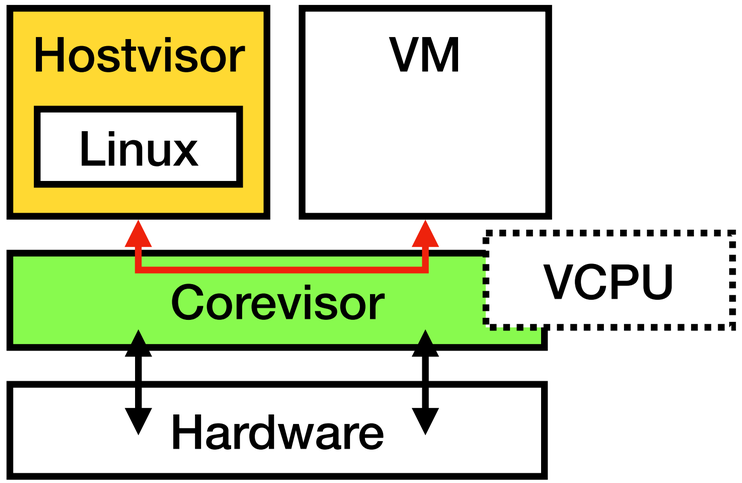
Memory Protection

- HypSec tasks hostvisor to provide memory allocation
  - Use identity mapping in hostvisor's NPT



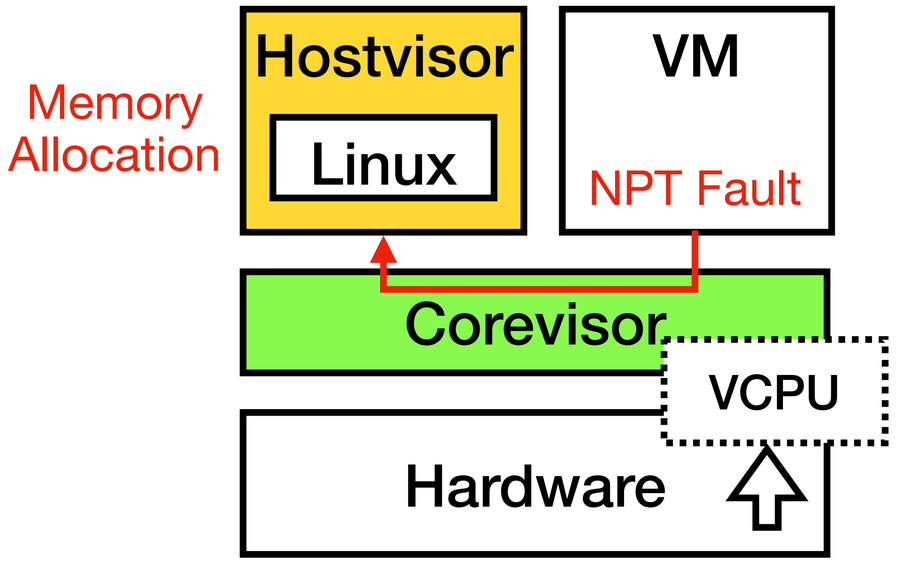
CPU Protection

- Corevisor ensures hostvisor cannot access VM CPU data on the hardware
  - Interpose switches between the VMs and hostvisor
  - Context switch VM's CPU registers to corevisor's private memory



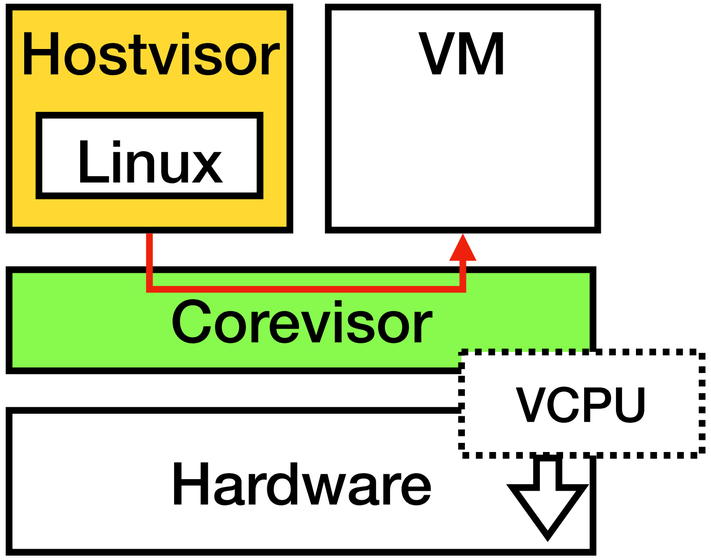
Example: Handling VM NPT Page Fault

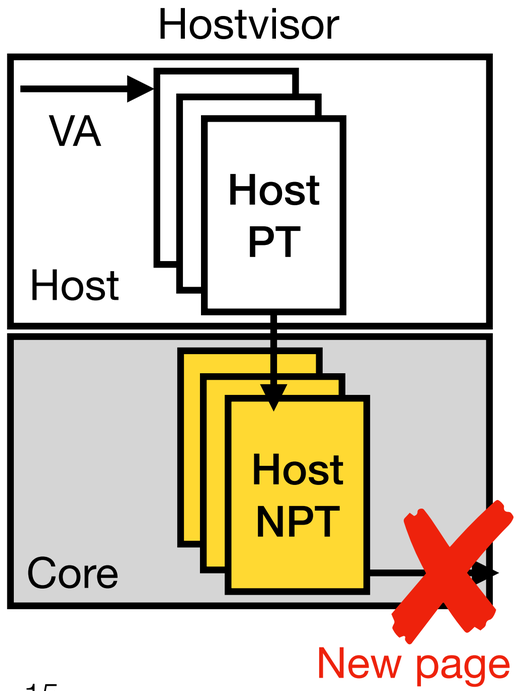
- Corevisor controls the hardware to trap NPT page faults to itself
  - Corevisor saves VM CPU from the hardware and switches to hostvisor for memory allocation

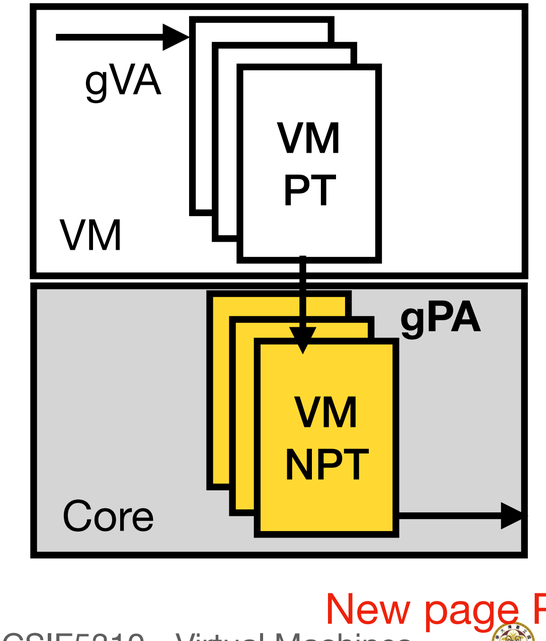


Example: Handling VM NPT Page Fault

- Corevisor unmaps newly allocated page from NPT and maps it to VM's NPT
  - KCore restores VM CPU to the hardware before entering VM







Deprivileging Hypervisors: Limitations

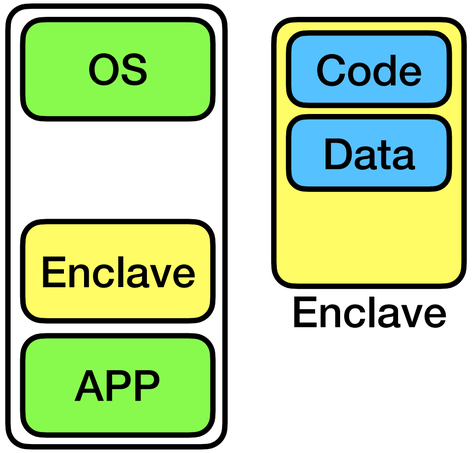
- CloudVisor and HypSec provide strong VM protection but comes with performance overhead:
  - CloudVisor inherits the performance overhead from nested virtualization
  - HypSec requires modification in VM kernel to enable memory sharing — why this matters?
- CloudVisor, HypSec, and pKVM, all rely on a software TCB to protect VMs
  - What if the TCB itself contains bugs?

Hardware Approach for Improving VM Security

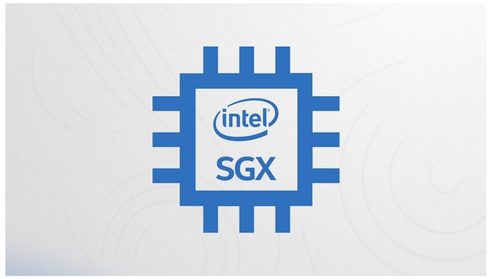
- Modern hardware provides security extensions to protect VM data
  - Intel Software Guard Extension (SGX)
  - AMD Secure Encrypted Virtualization (SEV)
- Leverage these hardware extensions to protect VM security
  - Haven [OSDI 14], Fidelius [HPCA 18]

Intel SGX

- SGX is a new instruction set to create ***enclave***, an encrypted area for running user-level code and data
- SGX protects the integrity and confidentiality of the enclave from:
  - Other applications
  - OS or hypervisor
- The enclave is only decrypted inside the CPU
  - Protect against physical RAM snooping



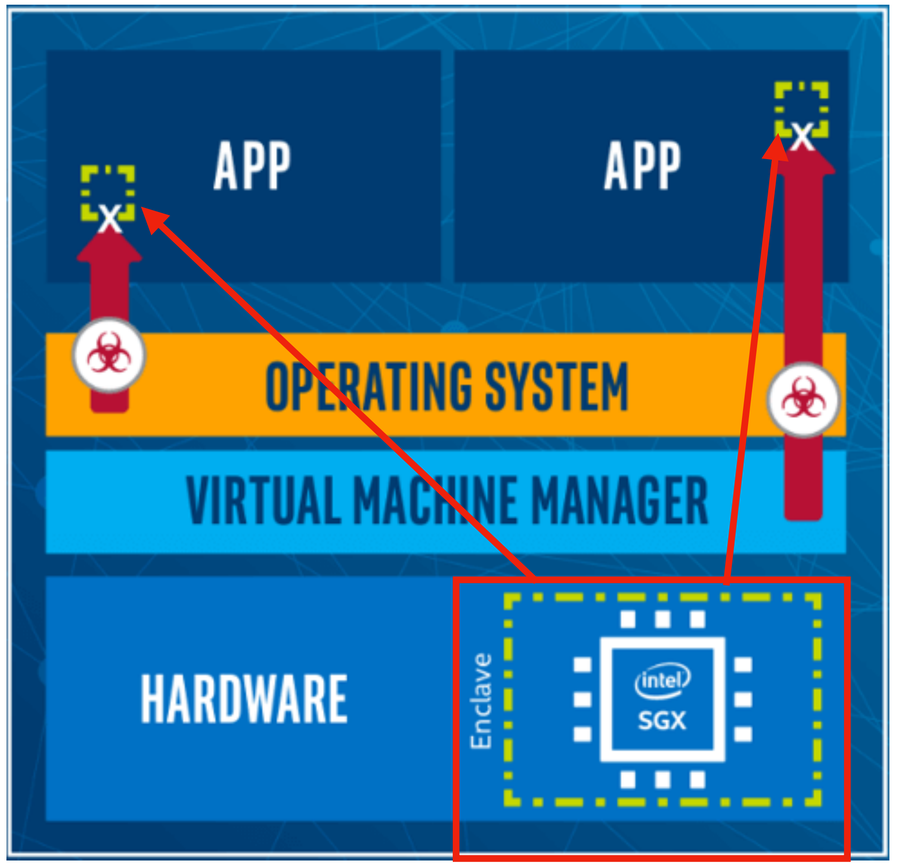
Intel SGX



Azure confidential computing

Solutions on Azure for Intel SGX

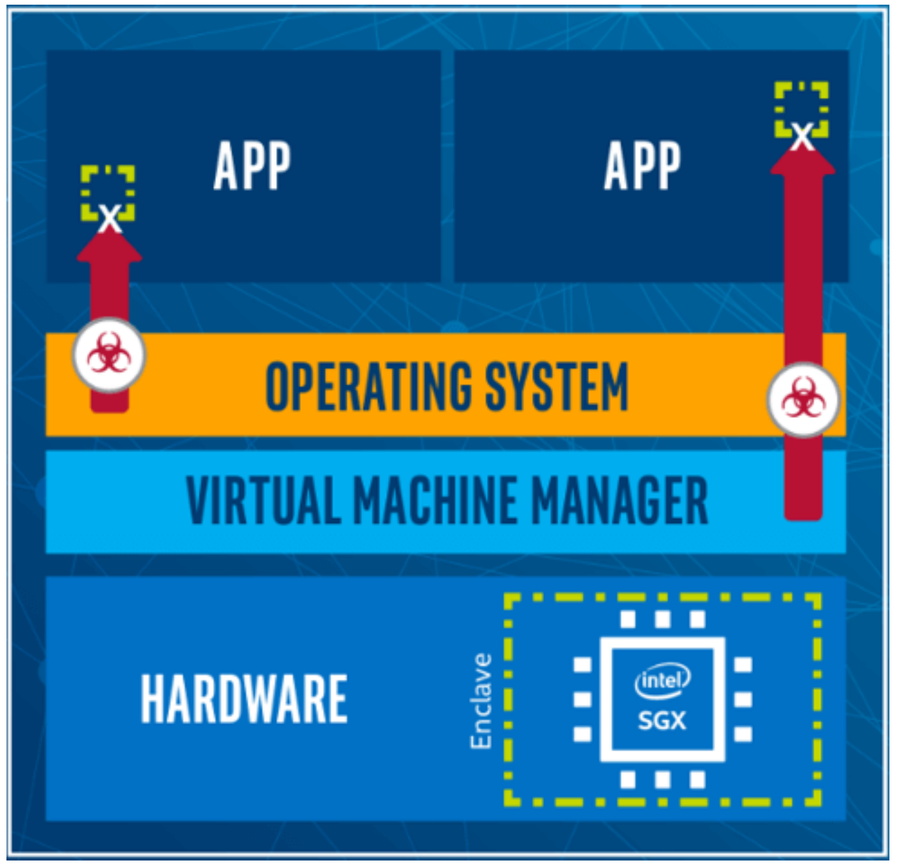
Article • 08/23/2022 • 3 minutes to read • 5 contributors



TEE

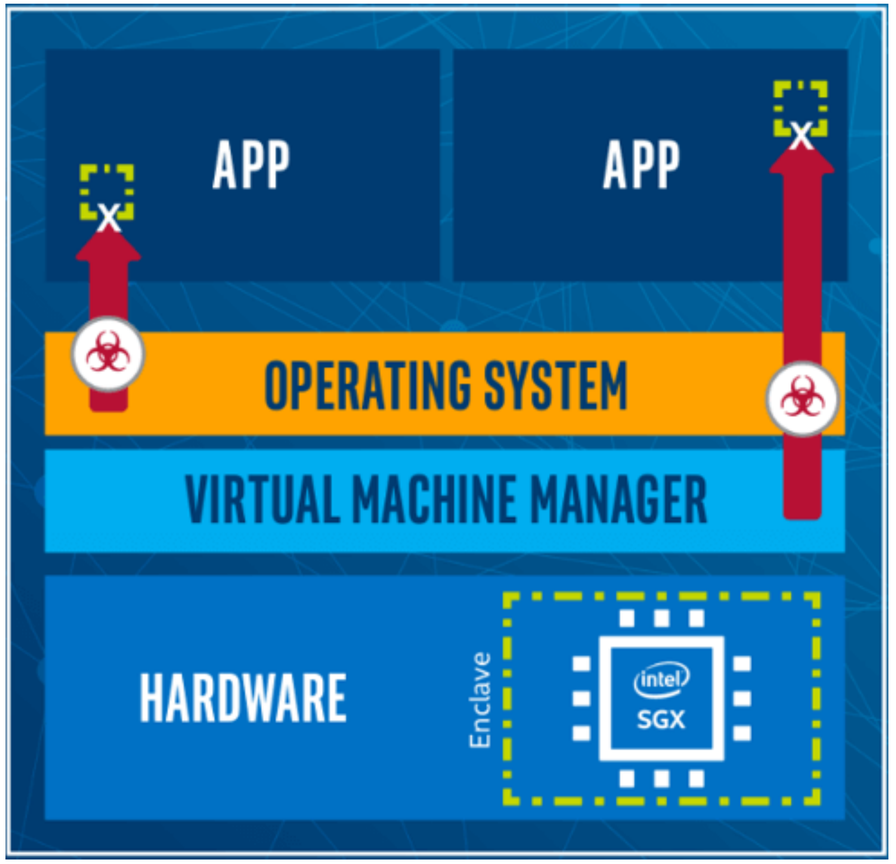
Intel SGX: Limitations

- Memory Limitations
  - Size limit of protected memory in enclave: 128 MB
  - In the initial SGX hardware, all memory must be committed at enclave build time
    - Developers to predict max heap and stack size
    - Additional code modules cannot be dynamically loaded
    - Page protection settings cannot be dynamically changed
  - SGX2 addresses the above limitations
    - Support larger memory size in enclaves
    - Allows dynamic memory management



Intel SGX: Limitations

- Security & Feature Limitations
  - Vulnerable to side-channel attacks
  - Cannot run a VM or OS kernel in the enclave
  - Enclaves interact with the untrusted OS for system calls; any issues? Have you heard of Iago attacks?



Hardware Approach for VM Protection: Haven

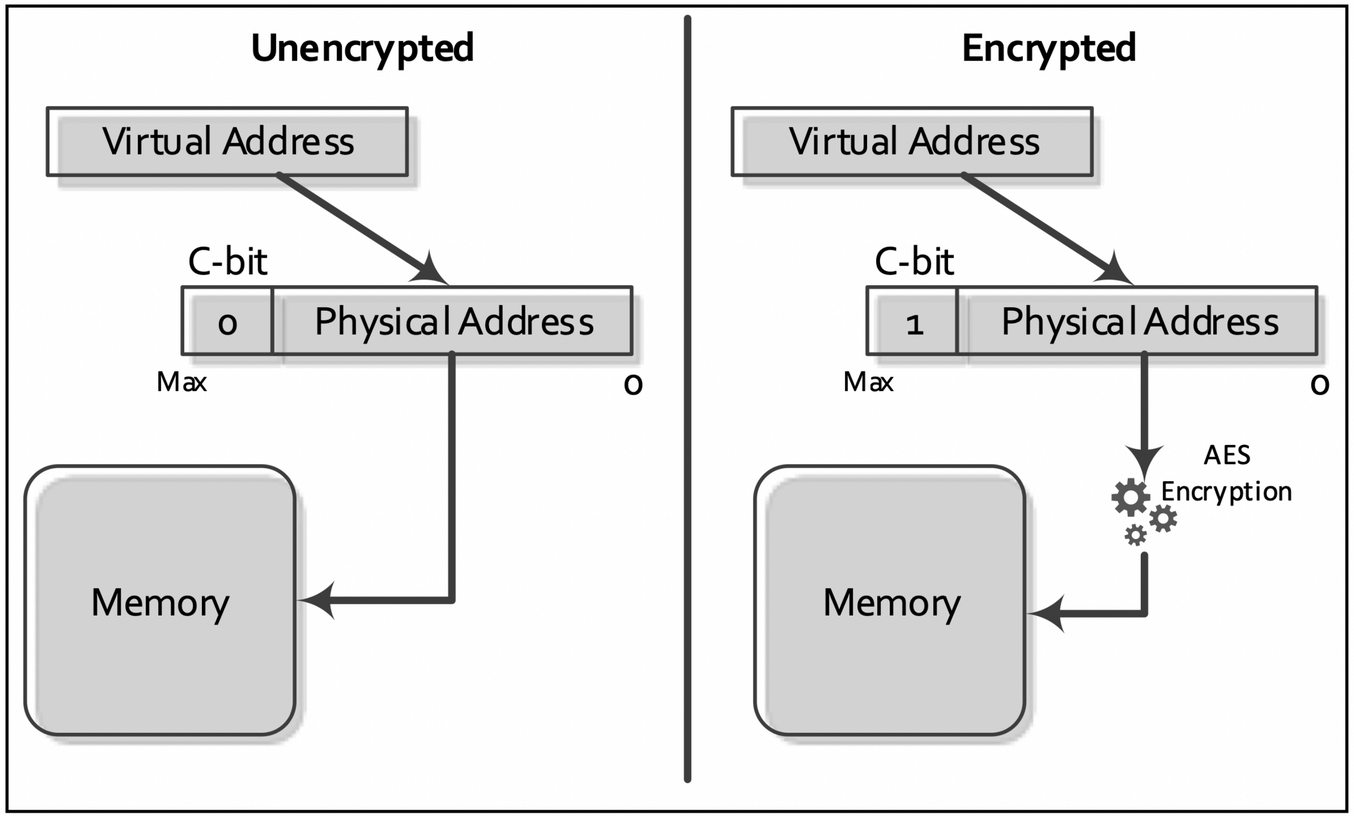
- **Goal:** protect confidentiality and integrity of the VM application against a malicious OS and hypervisor
- **Threat model:** assume attackers fully control the hypervisor or OS; assume attackers can physically access the machine
- **Mechanism and Policy:** run the trusted application in an SGX enclave
  - Incorporate an untrusted OS kernel to provide services to the enclave

[1] Shielding Applications from an Untrusted Cloud with Haven, Andrew Baumann, et all, OSDI 14

AMD SEV

- AMD introduces SEV in 2016 to allow the VM to protect itself from the hypervisor
- Employs an encryption based approach to protect VM memory and CPU state
  - Use a unique per VM AES key to perform encryption (associated with ASID)
  - Is it possible for the hypervisor to exploit ASID assignments to compromise VMs?
- SEV includes:
  - Secure Memory Encryption (SME): encrypts VM memory using the per-VM key
  - Encrypted State (ES): encrypts the VM's CPU registers when a VM stops running
  - Secure Nested Paging (SNP): ensures the integrity of the VM memory contents

AMD SEV



From: https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

https://www.google.com/url?sa=i&url=https://www.phoronix.com/news/AMD-SEV-Local-Migration&psig=AOvVaw2pdexyldHR3Gwk3eRl27Bn&ust=1677143837310000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCNjBhOflqP0CFQAAAAAdAAAAABAE

AMD SEV: Limitations

https://www.google.com/url?sa=i&url=https://www.phoronix.com/news/AMD-SEV-Local-Migration&psig=AOvVaw2pdexyldHR3Gwk3eRl27Bn&ust=1677143837310000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCNjBhOflqP0CFQAAAAAdAAAAABAE

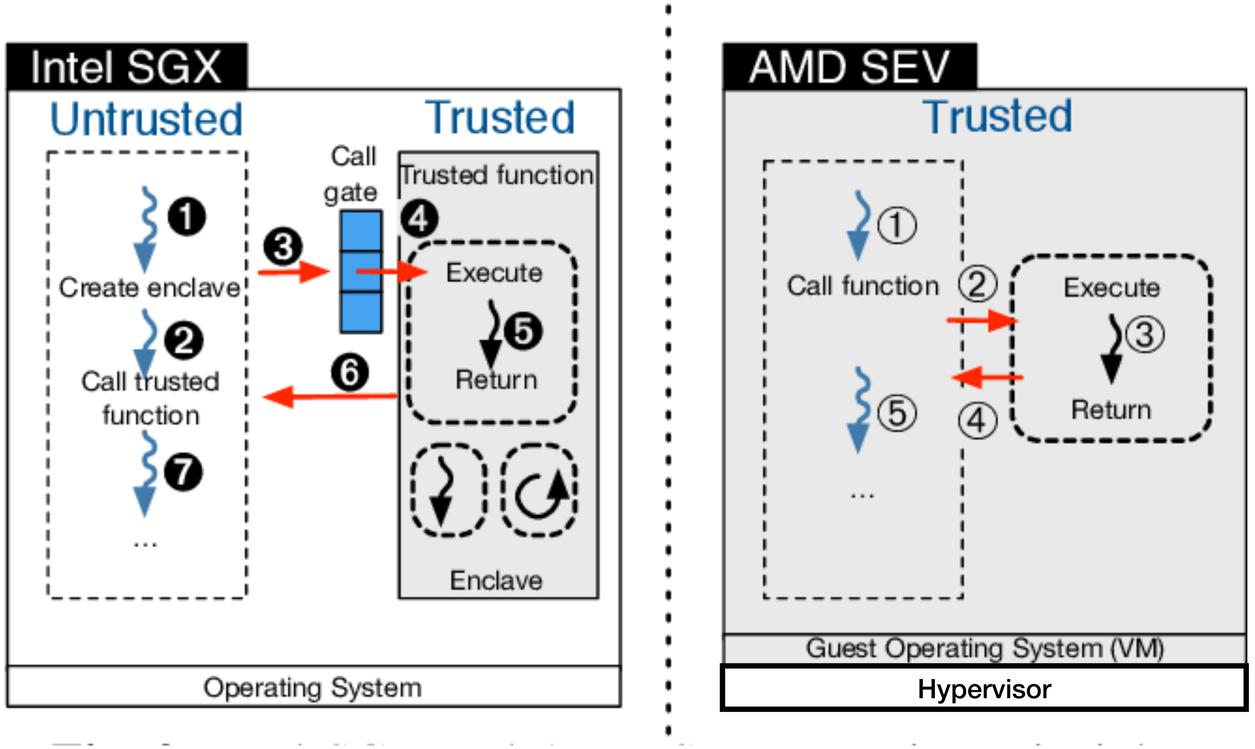
- Security Limitations
  - Also vulnerable to side-channel attacks

30TH USENIX SECURITY SYMPOSIUM

CIPHERLEAKS: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel

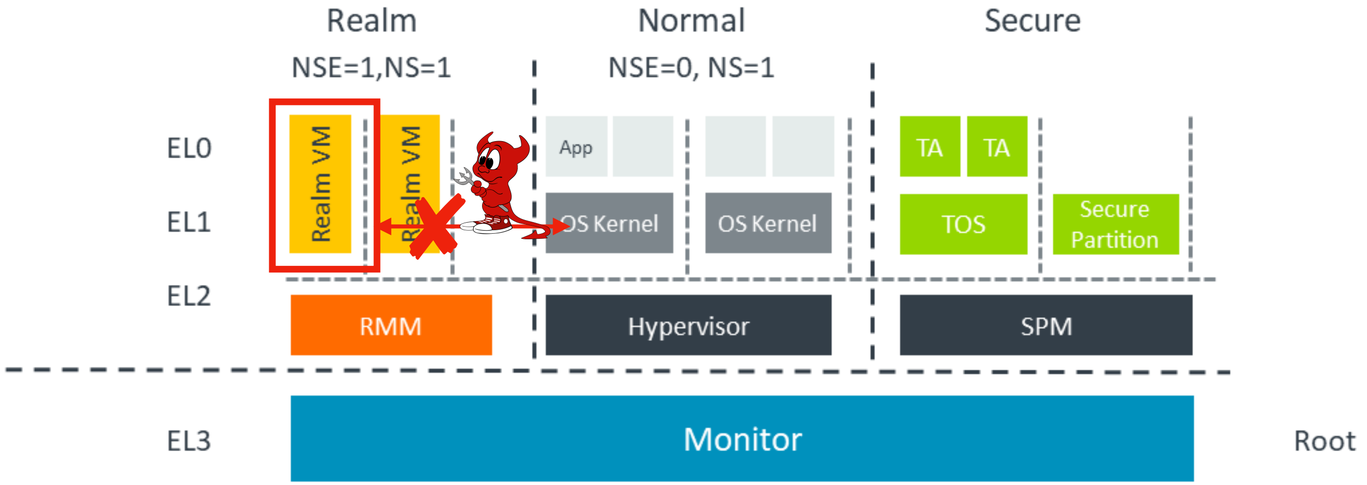
Confidential VMs

Intel SGX vs AMD SEV



Modified from the paper: Security, Performance and Energy Trade-Offs of Hardware-Assisted Memory Protection Mechanisms

Arm Confidential Compute Architecture (CCA)



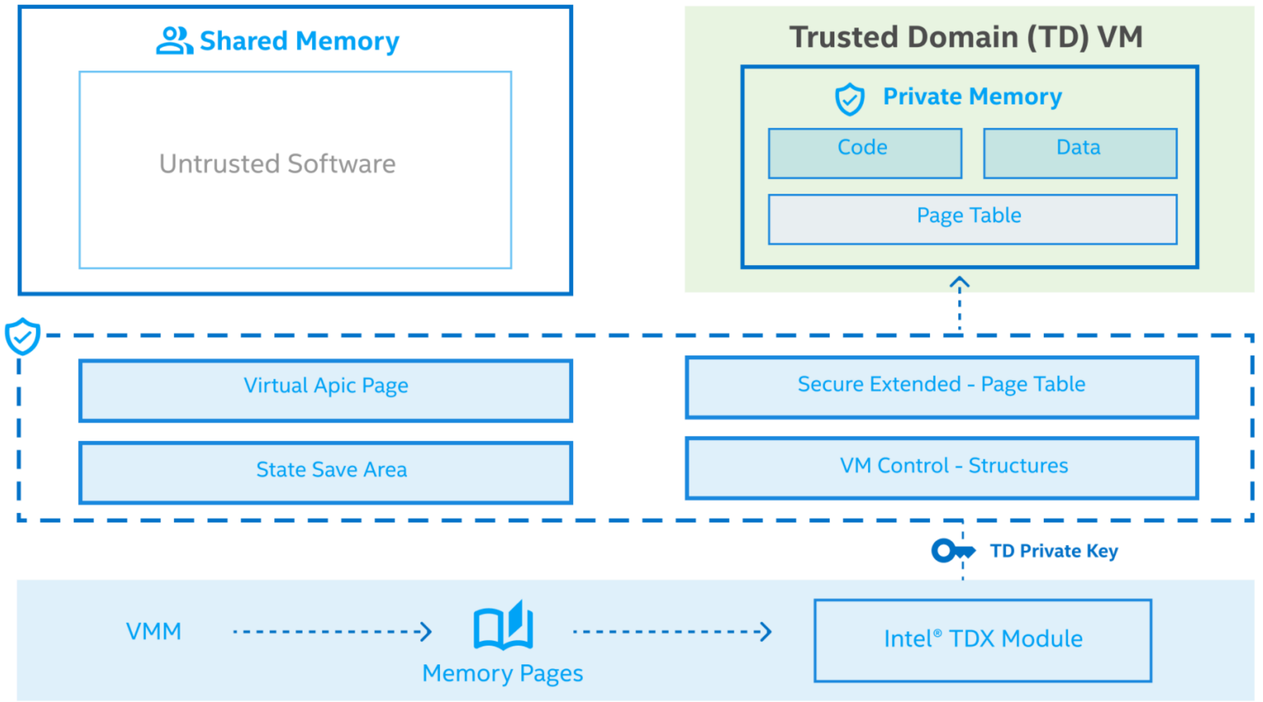
Arm Confidential Compute Architecture (CCA)

- Arm introduced the CCA for the Arm-v9 architecture in 2021
  - No actual implementation in production
- CCA ensures that a malicious hypervisor cannot access a secure VM's data in memory and CPU
- Rely on a formally verified RMM [1] to enforce the access control protection policy

[1] Design and Verification of the Arm Confidential Compute Architecture, Xupeng Li, Xuheng Li, Christoffer Dall, Ronghui Gu, Jason Nieh, Yousuf Sait, and Gareth Stockwell, Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2022), Carlsbad, CA, July 2022.

Intel Trust Domain Extensions (TDX)

- Goal: protect the confidentiality and integrity of Virtual Machines
- Intel TDX employs both encryption and access control for VM protection



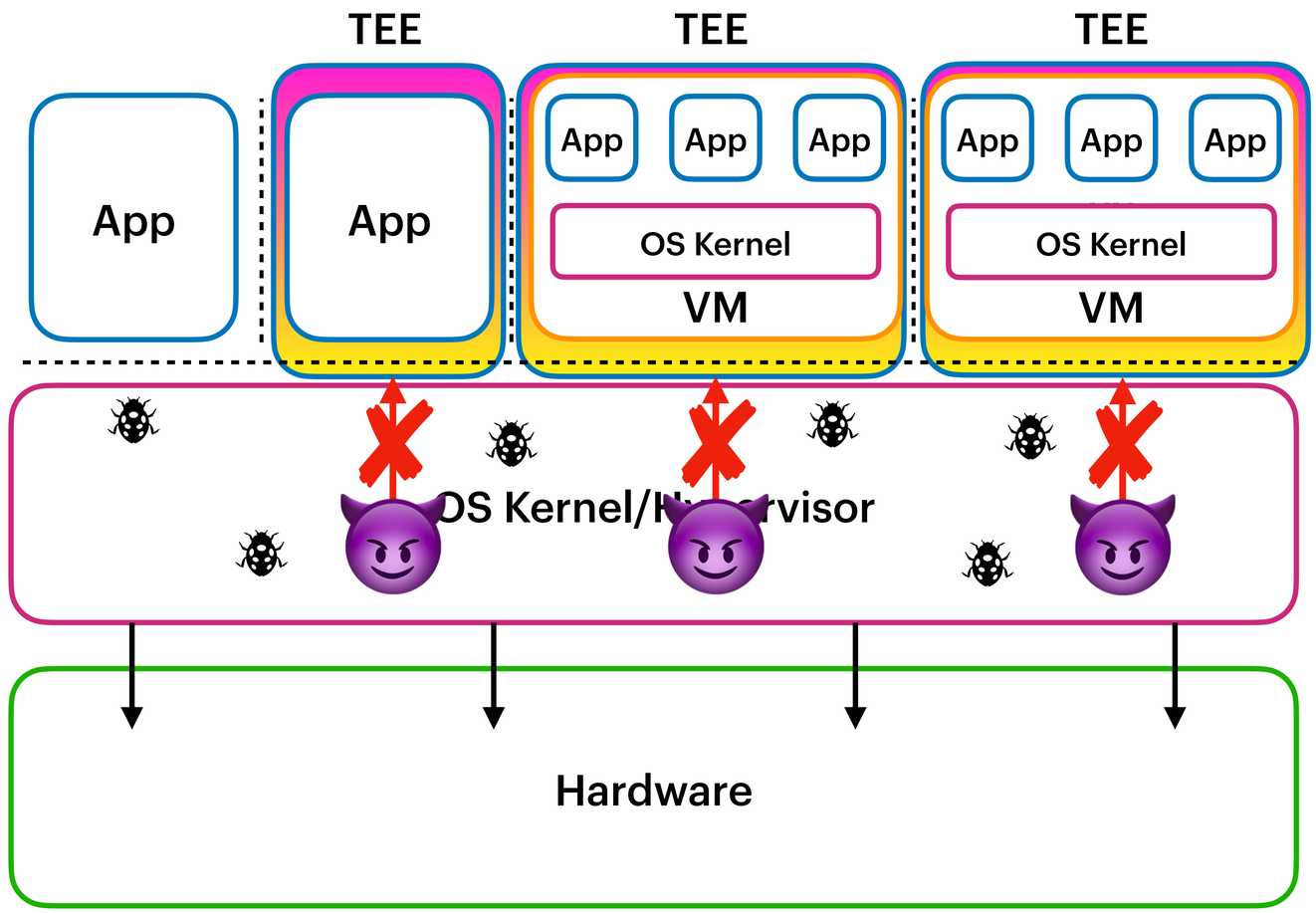
https://www.intel.com/content/dam/www/central-libraries/us/en/images/2023-09/tdx-diagram-16x9.png.rendition.intel.web.1648.927.png

Hardware Approach for VM Protection: Status Quo

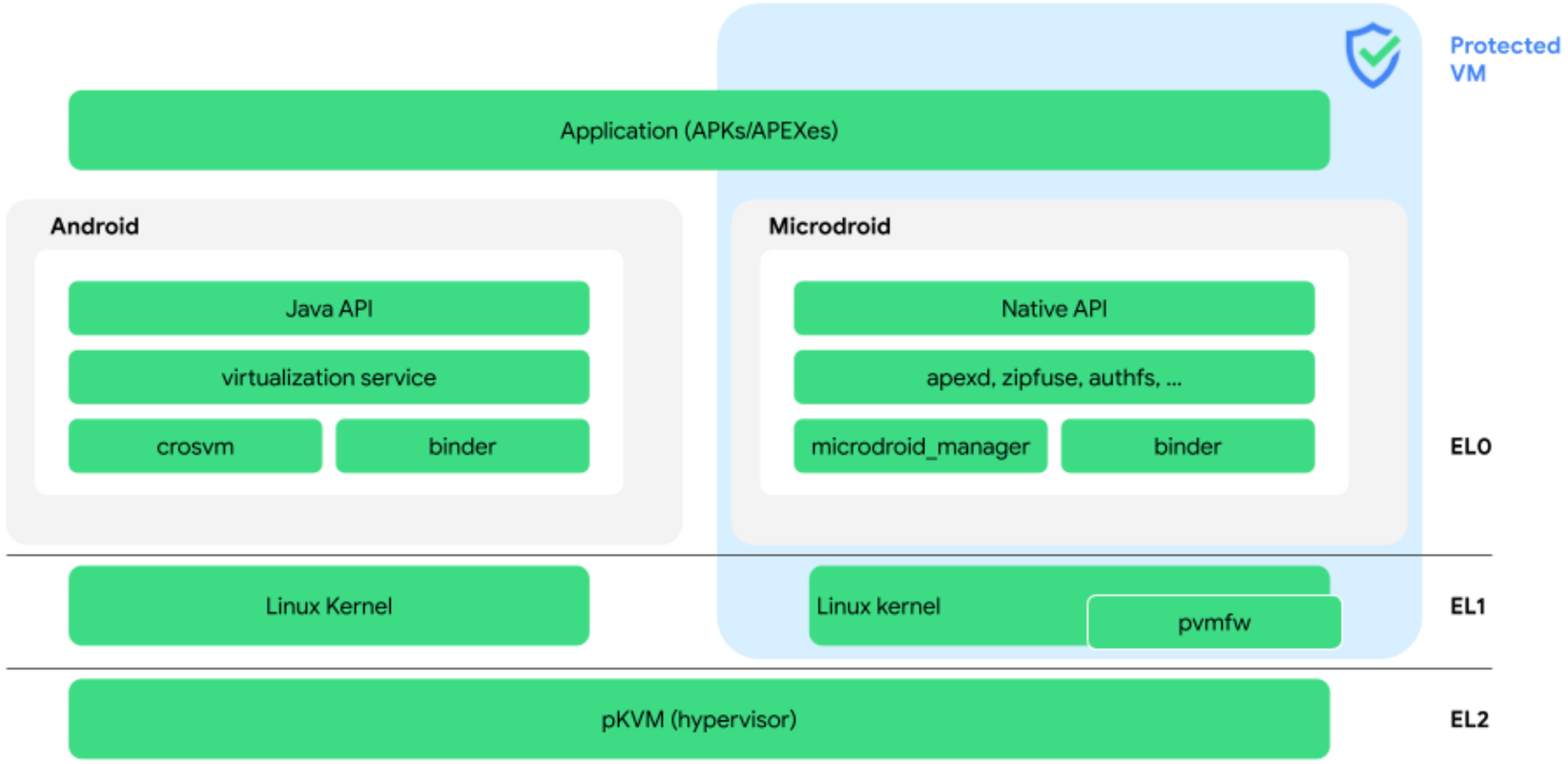
- Support for Intel SGX and AMD SEV has been provided by commodity software like Linux, KVM
- Past research focused on attacking or building secure systems on top of Intel SGX, AMD SEV, and Arm CCA

Confidential Computing

- Provides ***Trusted Execution Environment (TEEs)*** to protect user data
- Attackers that control the supervisor software (OS/hypervisor) ***cannot*** access the data/resources of the user in the TEE
  - Make sure the user data cannot be read or modified by the supervisor software
- Various approaches:
  - Encryption v.s. Access Control
  - Applications v.s. VM Protection (confidential VMs)



Google pKVM (protected KVM)



https://source.android.com/docs/core/virtualization?hl=zh-tw

Any Questions?