AMD

EXTENDING SECURE ENCRYPTED VIRTUALIZATION WITH SEV-ES

KVM FORUM - 2018

AGENDA

SEV Review

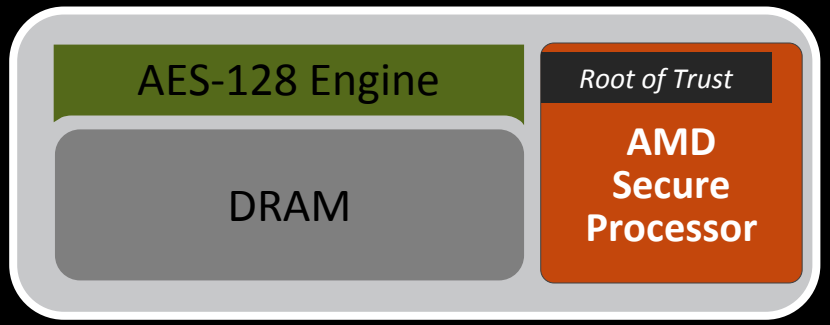
What is SEV-ES

How SEV-ES works

Patch Status

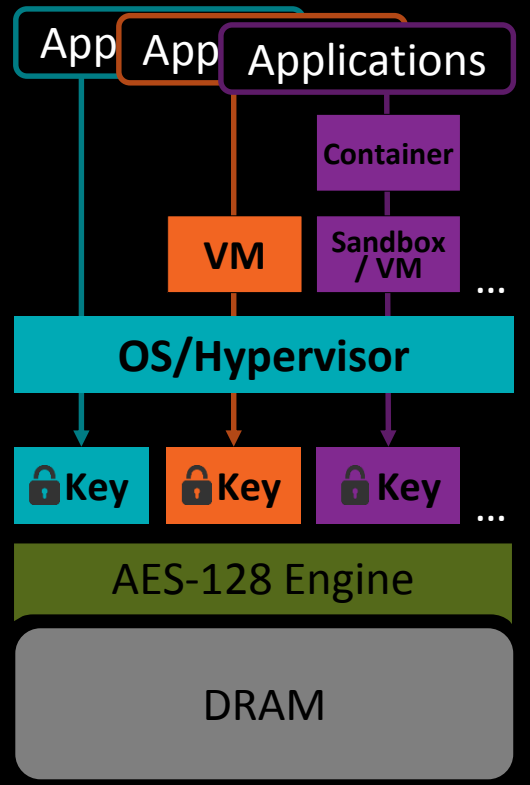
SEV REVIEW

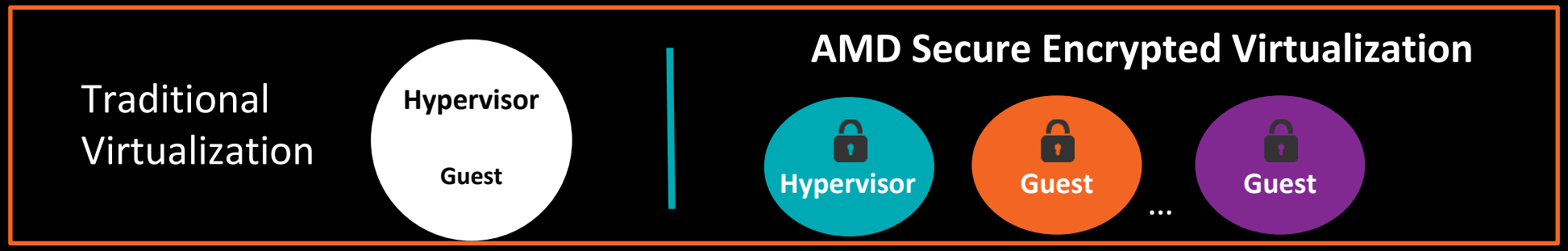
- AMD Secure Memory Encryption / AMD Secure Encrypted Virtualization
  - Hardware AES engine located in the memory controller performs inline encryption/decryption of DRAM
  - Minimal performance impact
    - Extra latency only taken for encrypted pages
  - No application changes required
  - Encryption keys are managed by the AMD Secure Processor and are hardware isolated
    - Not known to any software on the CPU



SEV REVIEW…

- Protects VMs/Containers from each other, administrator tampering, and untrusted Hypervisor
- One key for Hypervisor and one key per VM or VM/Sandbox with multiple containers
- Cryptographically isolates the hypervisor from the guest VMs
- Integrates with existing AMD-V™ technology
- System can also run unsecure VMs





SEV REVIEW…

- Linux Kernel
  - 4.15 - Run as an SEV guest
  - 4.16 - Launch an SEV guest
- EDK II / OVMF
  - 2018
- Qemu
  - 2.12
- Libvirt
  - 4.5

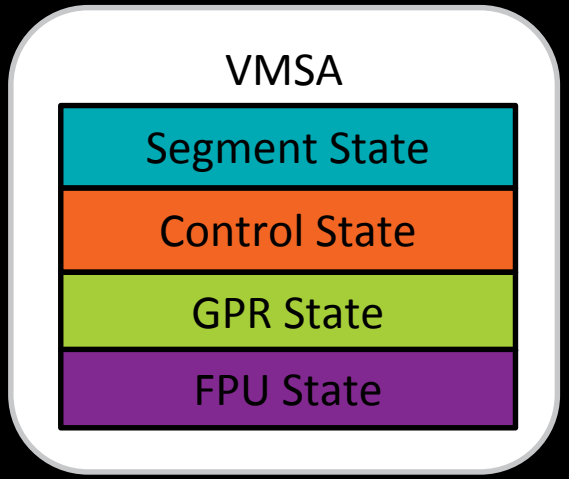
WHAT IS SEV-ES

- SEV: Secure Encrypted Virtualization
  - Provides encryption of guest memory

- SEV-ES: Secure Encrypted Virtualization – Encrypted State
  - Provides additional security above memory encryption
  - Guest register state is encrypted with guest encryption key and integrity protected
    - Only the guest can modify its register state
  - Guest must explicitly share register state with the hypervisor
    - Guest-Hypervisor Communication Block (GHCB)
  - Protects the guest register state from the hypervisor
    - Adds additional protection against VM state related attacks (exfiltration, control flow, rollback)

HOW SEV-ES WORKS

- Guest register state protection
  - Register state is initialized with known state (Initial Processor State)
  - Register state is encrypted and measured as part of the SEV LAUNCH process
  - Register state becomes part of the SEV LAUNCH measurement
  - Integrity check performed on each VMRUN
  - World switches now swap ALL register state
- VMCB under SEV-ES
  - Control Area (VMCA) and Save Area (VMSA) now separated
    - VMCA now points to VMSA
  - VMSA extended to save more state

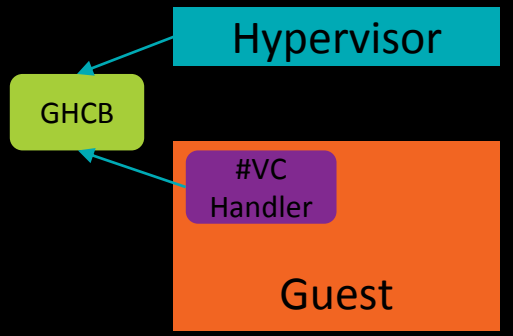


HOW SEV-ES WORKS…

- New VMEXIT types
  - Automatic Exit (AE)
    - Events that occur asynchronously with respect to guest execution (e.g. interrupts)
    - Events that do not require guest register state (e.g. HLT instruction)
  - Non-Automatic Exit (NAE)
    - Generates a VMM Communication Exception (#VC)
    - Guest determines what register state to share in the GHCB
    - Guest issues new VMGEXIT instruction which causes an AE with exit code 0x403
    - Guest updates register state with hypervisor supplied results
- New Control Register write traps
  - CR0 – CR15, EFER writes generate an AE after the control register has been modified
  - New value of the register is saved in EXITINFO1
  - Hypervisor can use this to track CR changes
  - Only supported for SEV-ES guests

HOW SEV-ES WORKS…

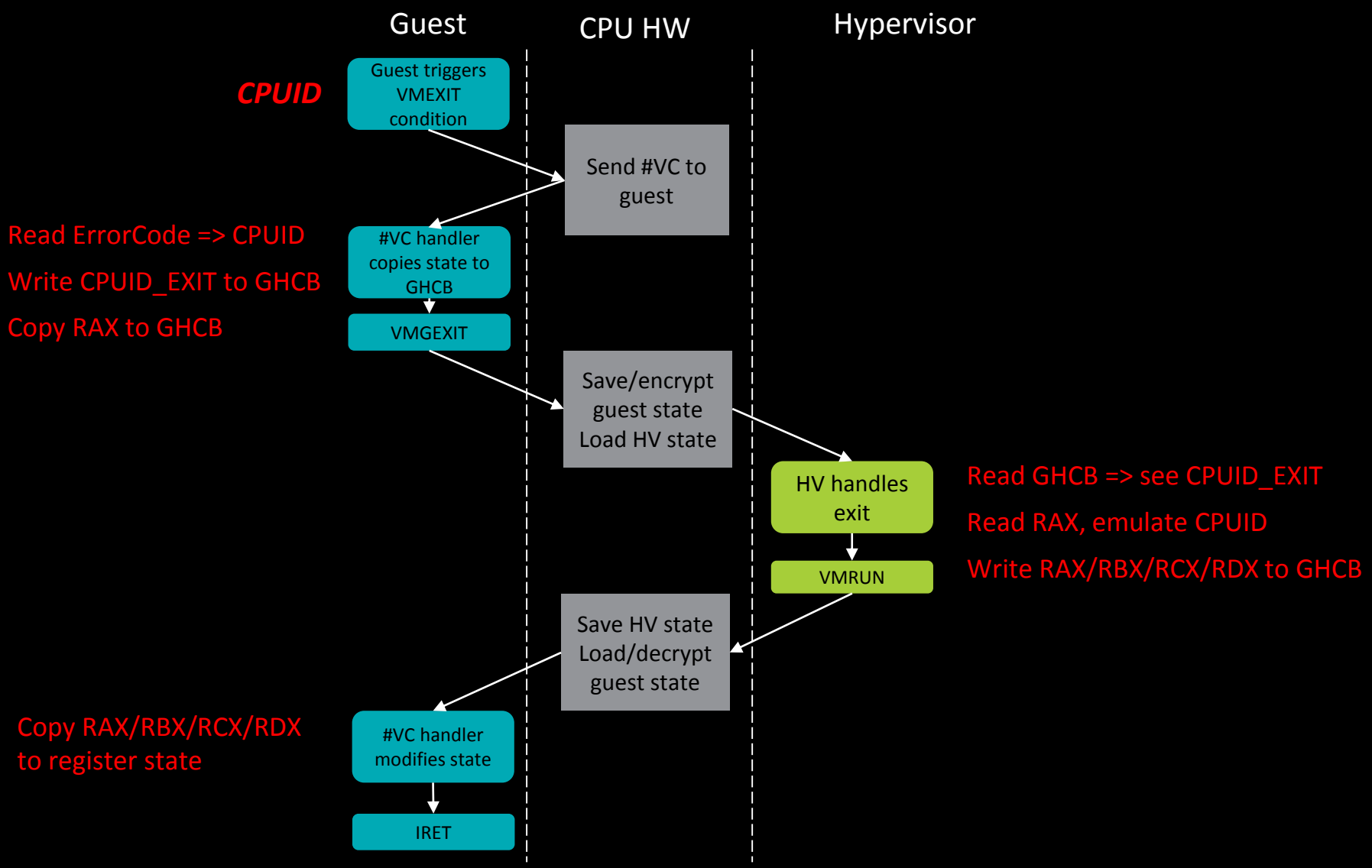
- Guest-Hypervisor Communication Block (GHCB)
  - Allows guest ←→ hypervisor communication of the state needed to satisfy the guest service request
  - Shared (un-encrypted) page between the hypervisor and the guest
    - Guest sets the physical address of the GHCB in an MSR
    - Hypervisor obtains guest physical address from the VMCB Control Area
  - GHCB specification (in process)
    - Define the format of the GHCB
    - Define a minimum set of supported NAE exits
    - Define the required state to be provided and returned on VMGEXIT
    - Define VMGEXIT definitions (SW_EXITCODE, SW_EXITINFO1, SW_EXITINFO2)
      - Re-use current SVM exit information where possible along with SEV-ES software-defined exit information
    - Define methods required for special scenarios
      - AP booting, NMI handling, etc.



HOW SEV-ES WORKS…

- New VMM Communication Exception (#VC)
  - Always thrown for an SEV-ES guest when an NAE event occurs
  - Error code is equal to the VMEXIT code of the NAE event
- #VC handler
  - Prepares the GHCB based on the error code
    - Copies required input register state to the GHCB
    - Sets SW_EXITCODE, SW_EXITINFO1, etc.
  - Issues VMGEXIT instruction to generate an AE event (REP VMMCALL)
  - Examines the GHCB on completion of VMGEXIT
    - Checks SW_EXITINFO1 for indication of success or failure
      - On failure, invoke requested exception (e.g. RDMSR of un-supported MSR)
    - Copies required output register state from the GHCB

HOW SEV-ES WORKS…



CHALLENGES

- Early Boot
  - Before CR4.PAE=1, all pages are private (encrypted)
    - Unable to mark the GHCB as shared (un-encrypted)
    - Need to avoid any NAE exits before marking the GHCB as shared
  - Encryption mask is normally determined using CPUID instruction – now causes a #VC
    - Requires the Hypervisor to communicate the encryption bit position to the guest
      - Hypervisor VMCB Control Area GHCB GPA field (offset 0x00a0) / Guest GHCB MSR (0xc0010130)
      - Format defined by GHCB specification

CHALLENGES…

- Early Boot
- SMP Support, NMI Support, others
  - Documented in the GHCB specification

CURRENT STATUS

- Currently have Proof-of-Concept patches for:
  - EDK2 / OVMF
    - Booting multi-vCPU guest through to UEFI shell / GRUB
  - Linux
    - Booting multi-vCPU guest
    - Need to address some early boot challenges
- GHCB Specification
  - Under review, available publicly

REFERENCES

- Links to the following reference material can be found at https://developer.amd.com/sev
  - White Papers & Specifications
    - Protecting VM Register State with SEV-ES Whitepaper
    - Guest Hypervisor Communication Block Specification
    - AMD64 Architecture Programmer's Manual Volume 2: System Programming
      - Sections 7.10, 15.34 and 15.35

DISCLAIMER

- The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.
- AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.
-