AMD SEV Update
Linux Security Summit 2018

David Kaplan, Security Architect

WHY NOT TRUST THE HYPERVISOR?

AMD

Guest Perspective

- Hypervisor is code I don’t control
- I can’t tell if the hypervisor is compromised
- I have obligations to protect my data
  - Financial data
  - Customer data
  - Health records
  - Etc.
- I want assurances of privacy

Hypervisor Perspective

- I need to convince more customers to use my services
- I don’t want to be able to see what customers are doing
- I want to provide assurances of privacy
- I want to limit my customer’s exposures to bugs

Security is a important concern for cloud growth

HARDWARE MEMORY ENCRYPTION - ATTACKS

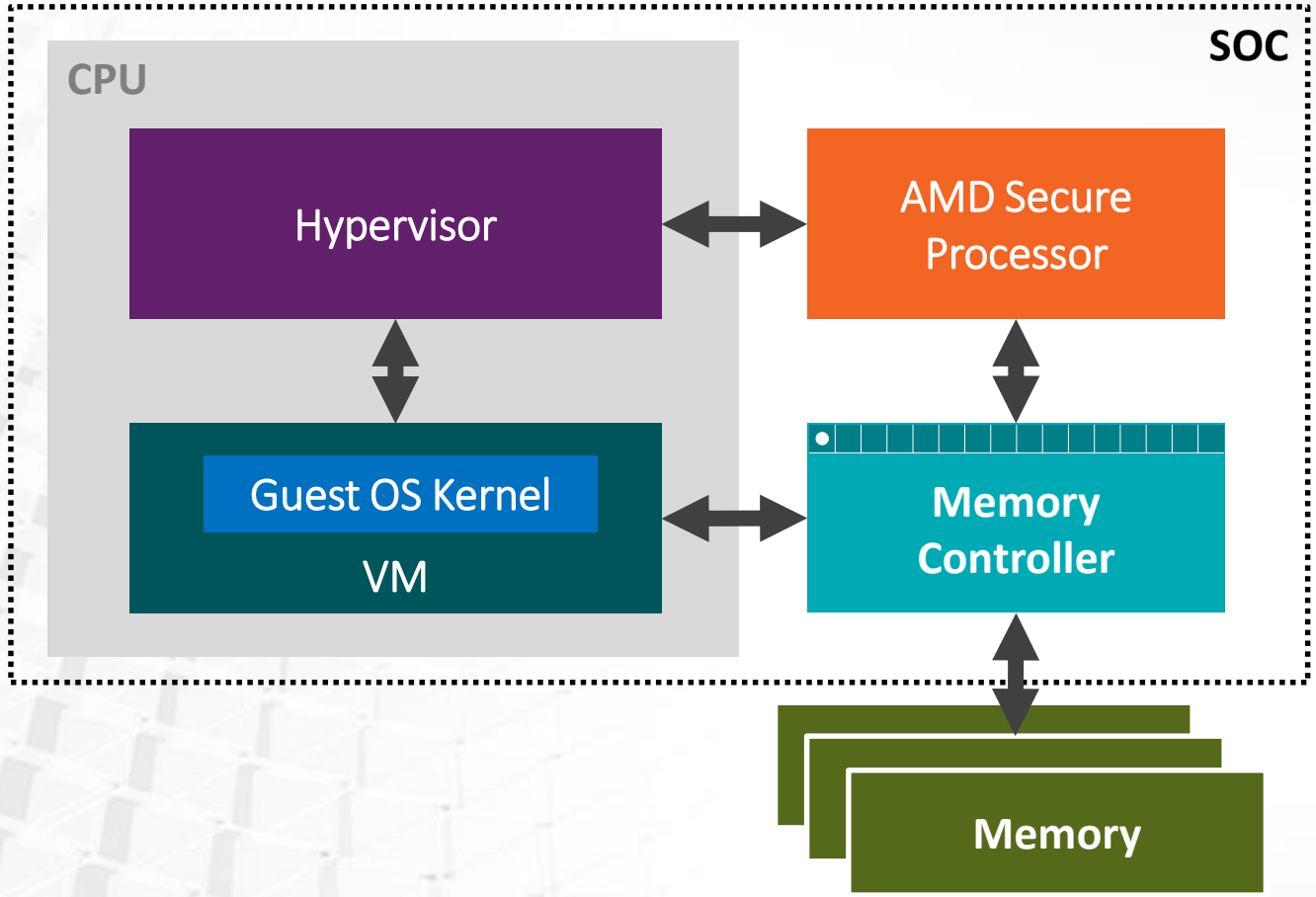
DEFENDED BY AMD SME + SEV

Physical Access Attacks	• Probe the physical DRAM interface • Install HW device that accesses guest memory • Freeze then steal DIMMs • Steal NVDIMMs
Admin Access Attacks	• Administrator scrapes memory of guest data areas • Administrator injects code into a guest VM • Hypervisor bug allows hosted guest to steal data from other guests

AMD HARDWARE MEMORY ENCRYPTION

SECURE ENCRYPTED VIRTUALIZATION (SEV) - COMPONENTS

- **AMD Secure Processor**
  - Manages keys during VM lifecycle
  - Runs SEV API firmware
  - https://developer.amd.com/wp-content/resources/55766.PDF
- **Hypervisor**
  - Calls SEV API as needed for VM management
  - Is protected with its own key
- **Guest OS**
  - Chooses which pages to encrypt via page tables



- **Example Development Environment**
  - KVM/QEMU Hypervisor
  - Linux Kernel for Guest
  - OVMF (UEFI) for Guest BIOS
  - SEV API (run in the AMD Secure Processor Firmware)

DEMO

<video>

SEV DEVELOPMENT UPDATE

AMD

Feature	Supported Version	Notes
SME (Host Encryption)	Linux 4.14	
SEV (Guest Support)	Linux 4.15	OVMF >= July 6, 2018
SEV (KVM HV Support)	Linux 4.16/QEMU 2.12	Libvirt 4.5
VirtIO support	Linux 4.15	Not including virtio-gpu
VirtIO-GPU support	Linux 4.19/QEMU 3.1	
SEV Guest Migration	-	Available on AMD Github
SEV Tool	-	
SEV Save & Restore	-	
SEV-ES Support	-	

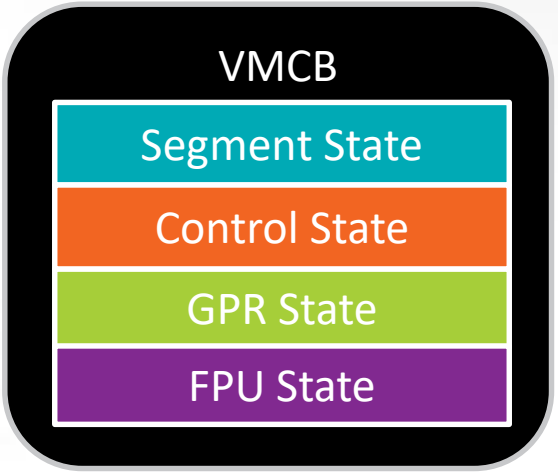
Completed

In Progress

- Ubuntu 18.04, Fedora 28, SLES 15 all are supported as SEV guests
- Fedora 28 and SLES 15 are supported as SEV hypervisors

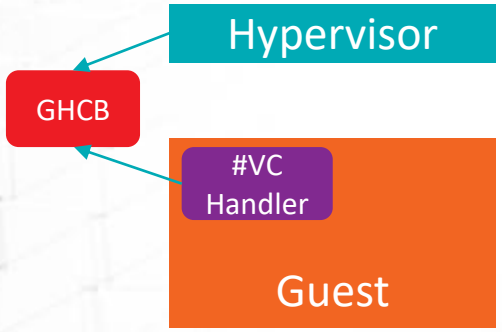
SEV-ES ARCHITECTURE AT A GLANCE

- World switches now swap ALL register state
  - Includes all segment registers, GPRs, FPU state (see Table B-4 in APM Vol2)
  - All register state is encrypted with the guest encryption key
  - Integrity value is calculated and stored in a protected page



- The guest is notified by a new exception (#VC) when certain events occur
  - The guest decides what state (if any) to share with the HV
  - The guest invokes the HV to perform the required tasks
  - The guest updates its state based on the output from the HV

- The guest and HV use a special structure to communicate
  - Guest-Hypervisor Communication Block (GHCB)
  - Location set by guest, mapped as unencrypted memory page



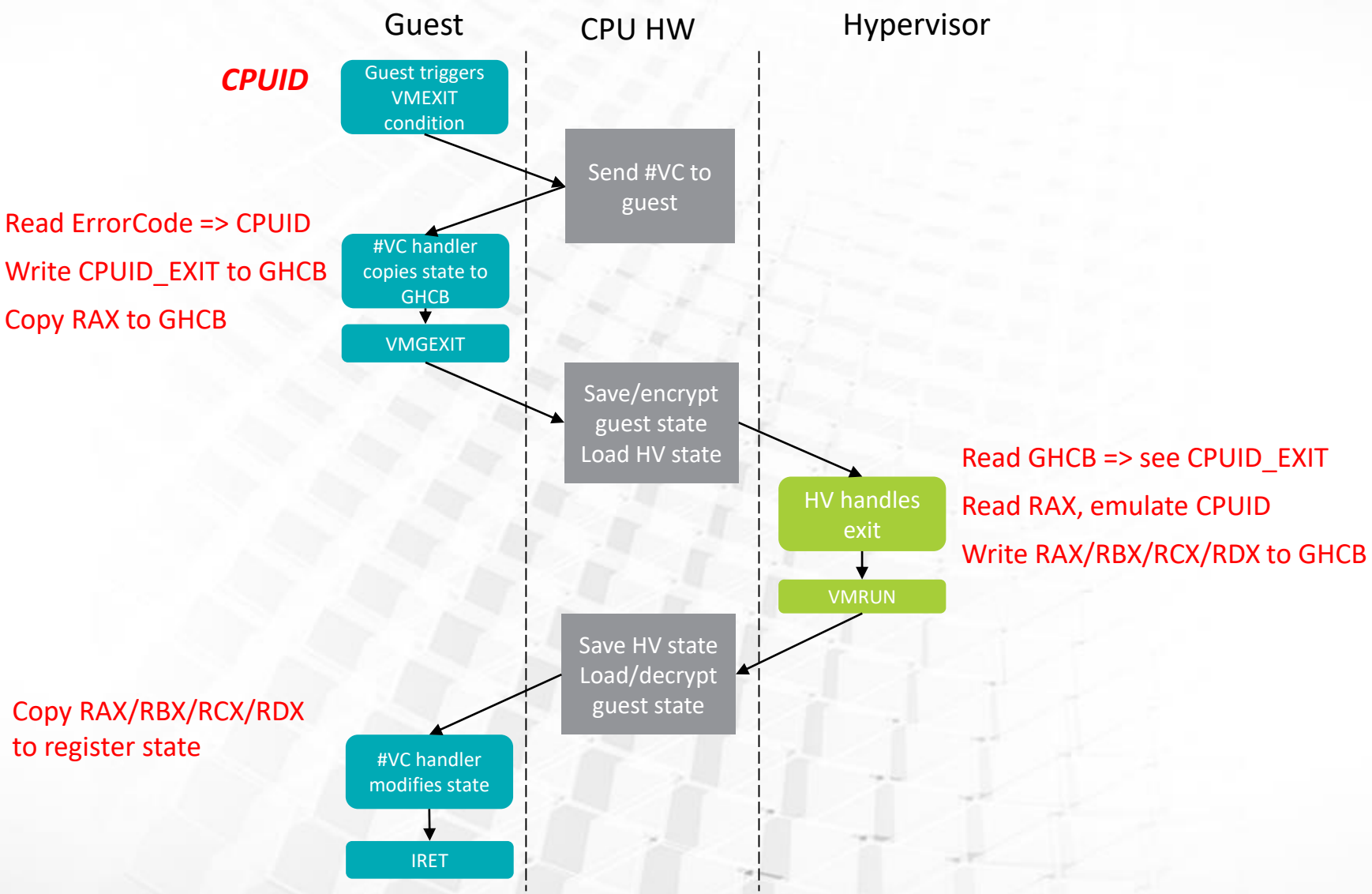
TYPES OF EXITS

AMD

- **Automatic Exits (AE)**
  - Events that occur asynchronously to the guest (e.g. interrupts)
  - Events that do not require exposing guest state (e.g. HLT)
  - Nested page faults not due to MMIO emulation
  - AE events save all state and exit to HV
  - Only action HV can do is just resume the guest w/o modifications
- **Non-Automatic Exits (NAE)**
  - All other exit events
  - NAE events cause a #VC instead of a VMEXIT
  - Guest handler may invoke the HV via VMGEXIT instruction

Code	Name	Notes	HW Advances RIP
52h	VMEXIT_MC	Machine check exception	No
60h	VMEXIT_INTR	Physical INTR	No
61h	VMEXIT_NMI	Physical NMI	No
63h	VMEXIT_INIT	Physical INIT	No
64h	VMEXIT_VINTR	Virtual INTR	No
77h	VMEXIT_PAUSE	PAUSE instruction	Yes
78h	VMEXIT_HLT	HLT instruction	Yes
7Fh	VMEXIT_SHUTDOWN	Shutdown	No
8Fh	VMEXIT_EFER_WRITE_TRAP	Write to EFER	Yes
90h-9Fh	VMEXIT_CR[0-15]_WRITE_TRAP	Write to CRx	Yes
400h	VMEXIT_NPF	Only if PFCODE[3]=0 (no reserved bit error)	No
403h	VMEXIT_VMGEXIT	VMGEXIT instruction	Yes
-1	VMEXIT_INVALID	Invalid guest state	-

NAE FLOW EXAMPLE



GHCB SPECIFICATION

AMD

- To attempt to standardize guest<->hypervisor interfaces, AMD has distributed a proposed GHCB Software Specification (see https://developer.amd.com/sev/)
- GHCB Specification defines:
  - Layout of GHCB memory page (4kb)
  - What #VC exceptions guests are expected to handle (super-set of all supported HV intercepts)
  - What values guests are expected to provide to the HV for each #VC
  - What the HV is expected to provide on each VMGEXIT
  - How SEV-ES guests are to be bootstrapped (including multi-vCPU environments)
  - NMI handling for SEV-ES guests
- Goal: To the greatest extent possible, provide a unified interface across all guest OS's and hypervisors that support SEV-ES

GHCB PROTOCOL

AMD

NAE Event	State to Hypervisor	State from Hypervisor	Notes
MSR_PROT (RDMSR)	RCX SW_EXITCODE=0x7c SW_EXITINFO1=0 SW_EXITINFO2=0	RAX RDX	
CPUID	RAX RCX XCR0 (for RAX=0xd) SW_EXITCODE=0x72 SW_EXITINFO1=0 SW_EXITINFO2=0	RAX RBX RCX RDX	XCR0 is only required to be supplied when a request for CPUID 0000_000D is made.

See proposed GHCB document for more details

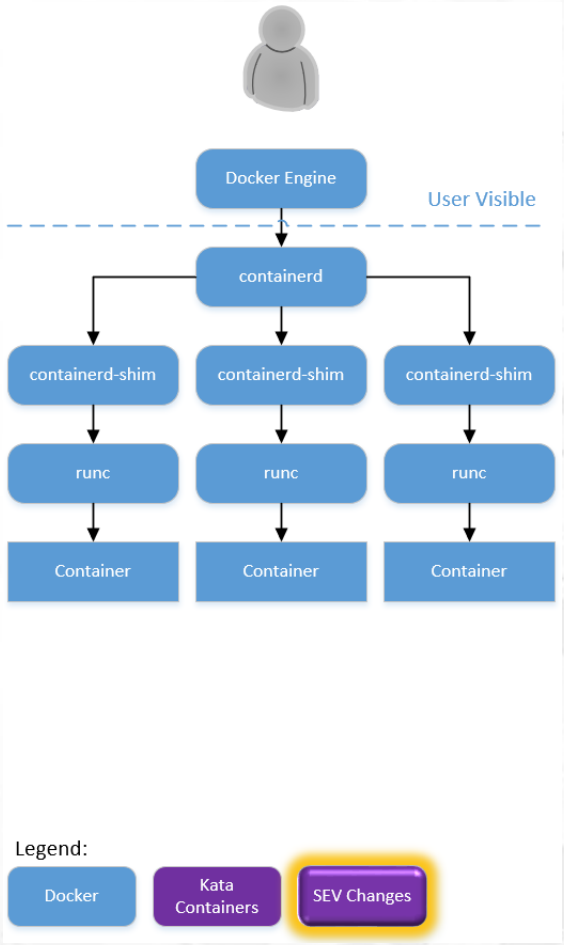
FEEDBACK SO FAR

- GHCB Specification is still undergoing review, comments are welcome!
- Comments:
  - GHCB memory layout matching VMCB layout -- Keeps hypervisor logic uniform, just changes where it gets values
  - Required registers for most intercepts -- Hypervisors are very uniform in this area
  - GHCB protocol negotiation -- Allow guest code to determine SEV-ES support at runtime
- Specific cases:
  - VMMCALL
  - AP startup
  - Debug support
  - SMM support

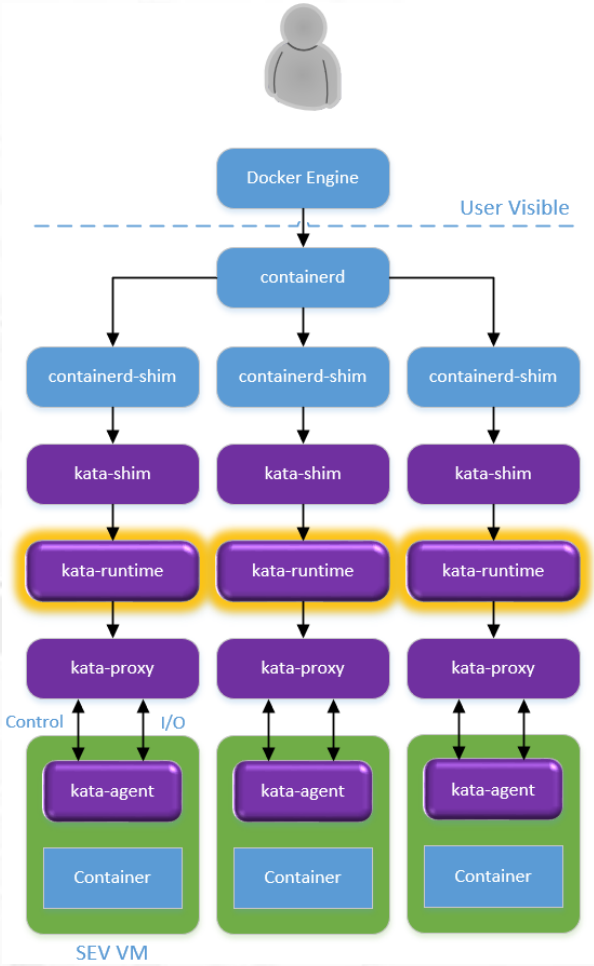
CONTAINERS WITH SEV

HYPERVISOR-BASED RUNTIME WITH SEV

Docker with runC (default)



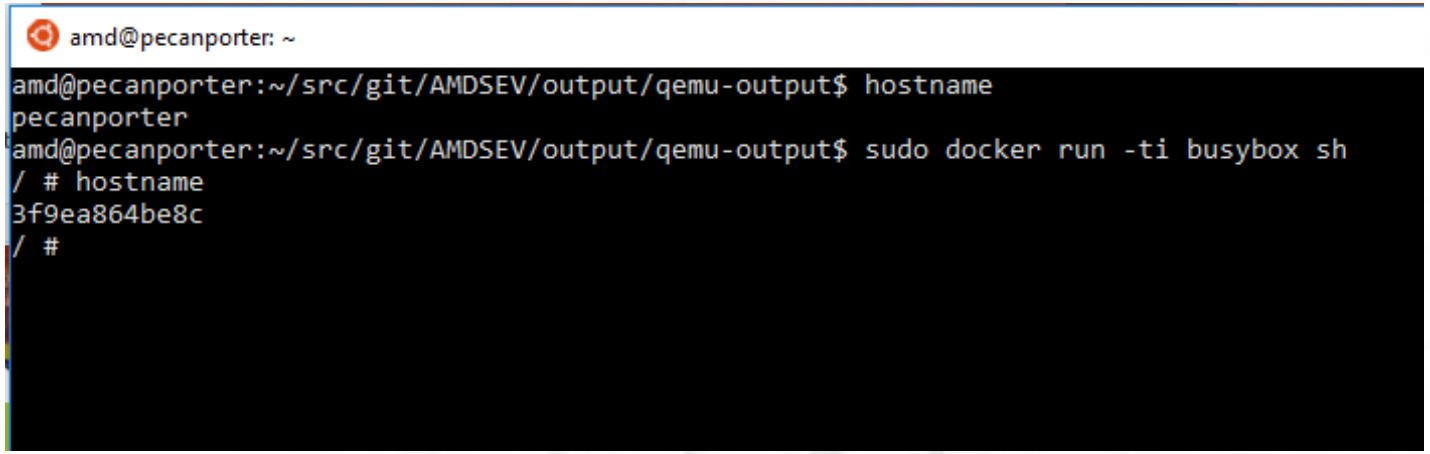
Docker with Kata Containers & SEV



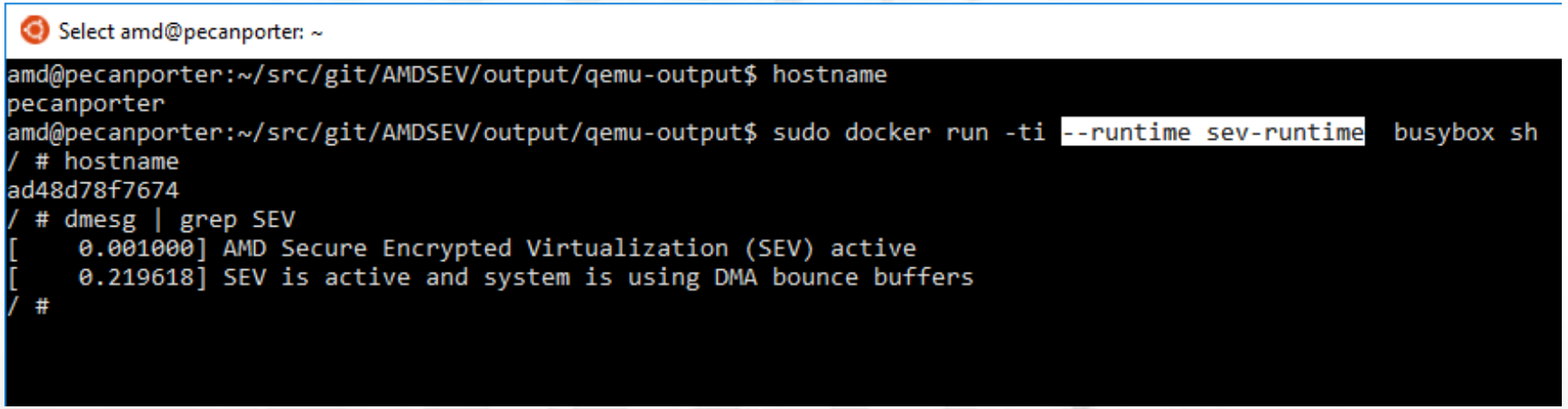
DEMO: SEV RUNTIME

DROP-IN REPLACEMENT FOR RUNC

Docker with runc (default)



Docker with SEV containers



CONCLUSION

AMD

- Upstream SEV work is progressing across multiple projects
- SEV-ES hardware is available today, software still in progress
- GHCB specification attempting to unify guest<->hypervisor para-virt interfaces
- Exploring other uses of SEV, such as with Kata Containers

- References:
  - AMD SEV info (https://developer.amd.com/sev/)
  - Github (with getting started scripts): https://github.com/AMDESE/AMDSEV
  - Kata Container prototype: https://github.com/AMDESE/AMDSEV/tree/kata
  - AMD64 Architecture Programmer's Manual Volume 2: System Programming (sections 7.10 and 15.34)
  - Secure Encrypted Virtualization Key Management


AMD

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION