

# Enhancing KVM for Guest Protection and Security

Jun Nakajima, Intel Corp.

# Intel Notice & Disclaimer

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting [www.intel.com/design/literature.htm](http://www.intel.com/design/literature.htm).

Intel, the Intel logo, {List the Intel trademarks in your document} are trademarks of Intel Corporation in the U.S. and/or other countries.

\*Other names and brands may be claimed as the property of others

© Intel Corporation.

# Agenda

- Security Implications of KVM
- “Secure Virtualization” for KVM
- Degree of Guest VMs Protection
- Proof of Concept
- Next Steps

# Security Implications of KVM



## KVM piggybacks on Linux systems

More attack surfaces, making guest more exposed

Guests could exploit host via user-space VMM



## Full access by user-space VMM:

Guest VM memory, vCPU states, etc.



## Full access by KVM/Linux Kernel:

**Any** guest VM memory, vCPU states, etc.

# Secure Virtualization for KVM

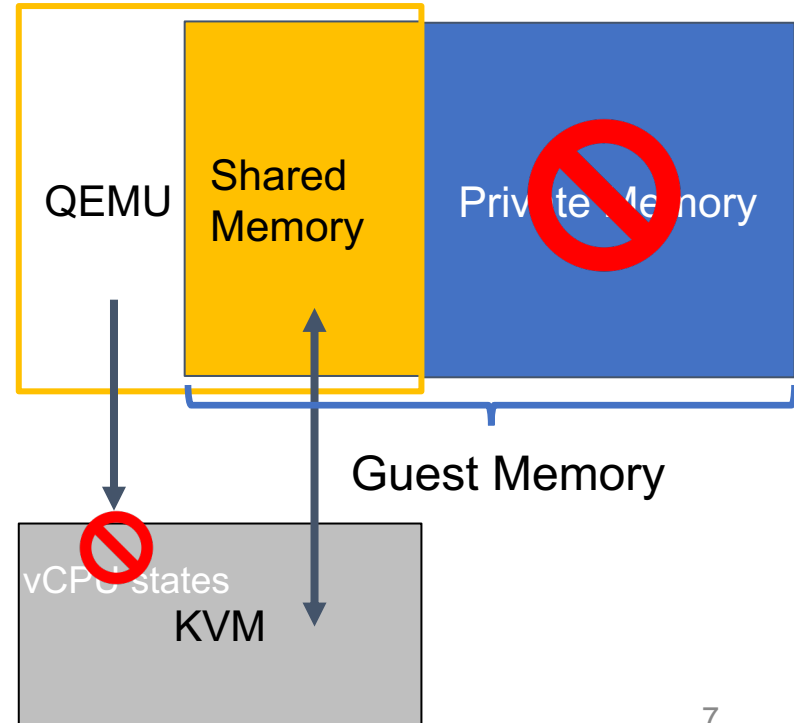
- Inspired by memory encryption technologies
- Protect guest VMs from a benign but vulnerable VMM
- Degree of “Protection” depends on what can be removed from TCB:
  - User-space VMM
  - Misc. kernel-space subsystems/drivers
  - Hypervisor

# Degree of Guest VMs Protection — what to remove from TCB

To protect potential attacks from	What's required
User-space VMM	Deny access to guest register state Deny access to guest <b>private</b> memory
Kernel-space ( <b>except KVM</b> ) + above	Remove direct mapping Deny access to guest <b>private</b> memory
Hypervisor (KVM) + above	Hardware-based security feature

# Secure Virtualization for KVM

- Guest specifies **shared** regions out of **private** memory
  - QEMU can access only **shared**
  - KVM may access **private** if in TCB
- KVM denies access to guest vCPU state by user-space VMM (e.g. QEMU)



# Modifications to Guest VM (Linux) when KVM is in TCB

- PV operations
  - Controls **shared** guest memory (GPA)
    - All **private** at boot time
  - Use swiotlb bounce buffer to force DMA operations in **shared** memory
  - Etc.
- (Optional) Hypervisor denies access by other kernel subsystems/drivers to guest **private** memory



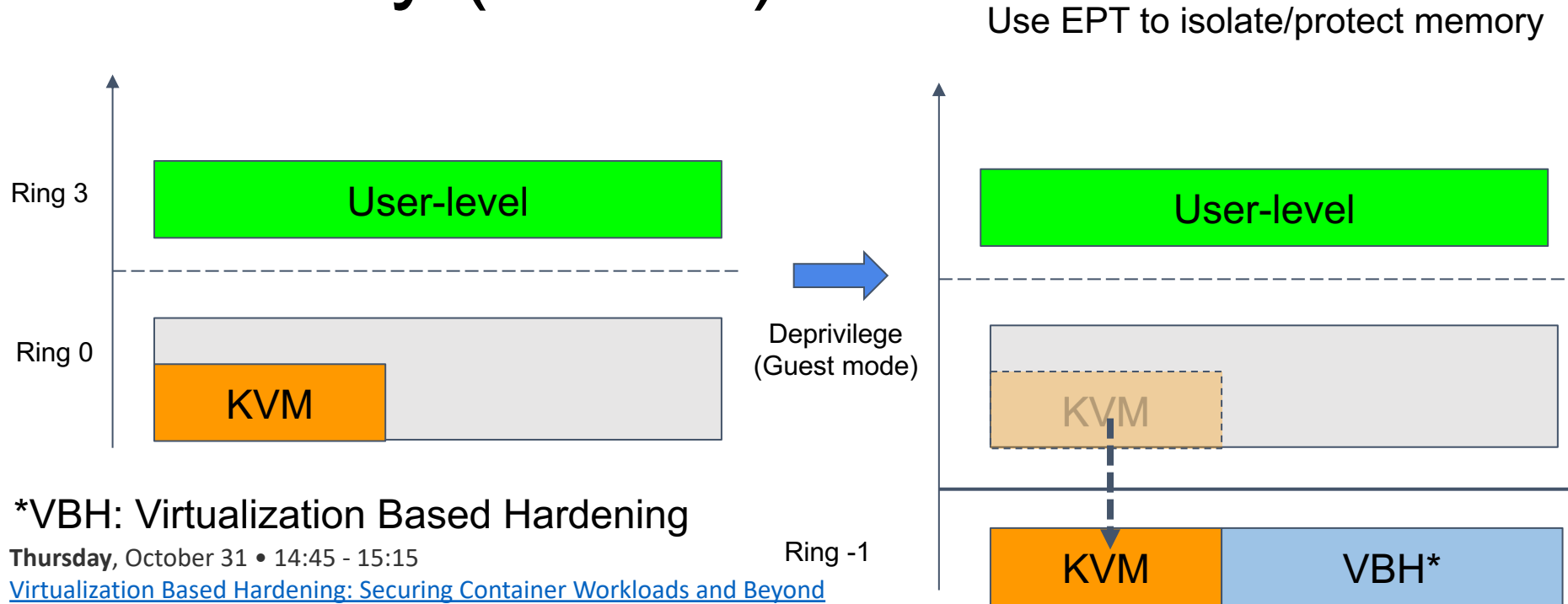
# Modifications to Guest VM (Linux) when KVM is **not** in TCB

- Additional PV operations
  - Avoid VM exiting operations that would require KVM to access guest VM memory
    - E.g. MMIO operations (replace them with Hypercall)
  - Or, handle them in guest VM via virtual exceptions
    - E.g. #VE (Virtual Exception) and emulate operations inside VM (and Hypercall)
- Useful even if KVM is in TCB

# Proof Of Concept

- KVM and other changes
  - Remove mapping from QEMU and kernel
  - “Ideal/optimal memory management for future VMs”, *Isaku Yamahata*, November 1 (11:30 - 12:00)
- Average CPU% overhead (virtio)\*:
  - 1.2% (1 VM), 1.3% (10+ VMs) for disk read, 1.1% (1 VM), 1.2% (10+ VMs) for disk write
  - 2.6% for network send (1 VM), 3.8% (10+ VMs)

# Deny Access to Guest Private Memory (Kernel)



\*VBH: Virtualization Based Hardening

Thursday, October 31 • 14:45 - 15:15

[Virtualization Based Hardening: Securing Container Workloads and Beyond](#)

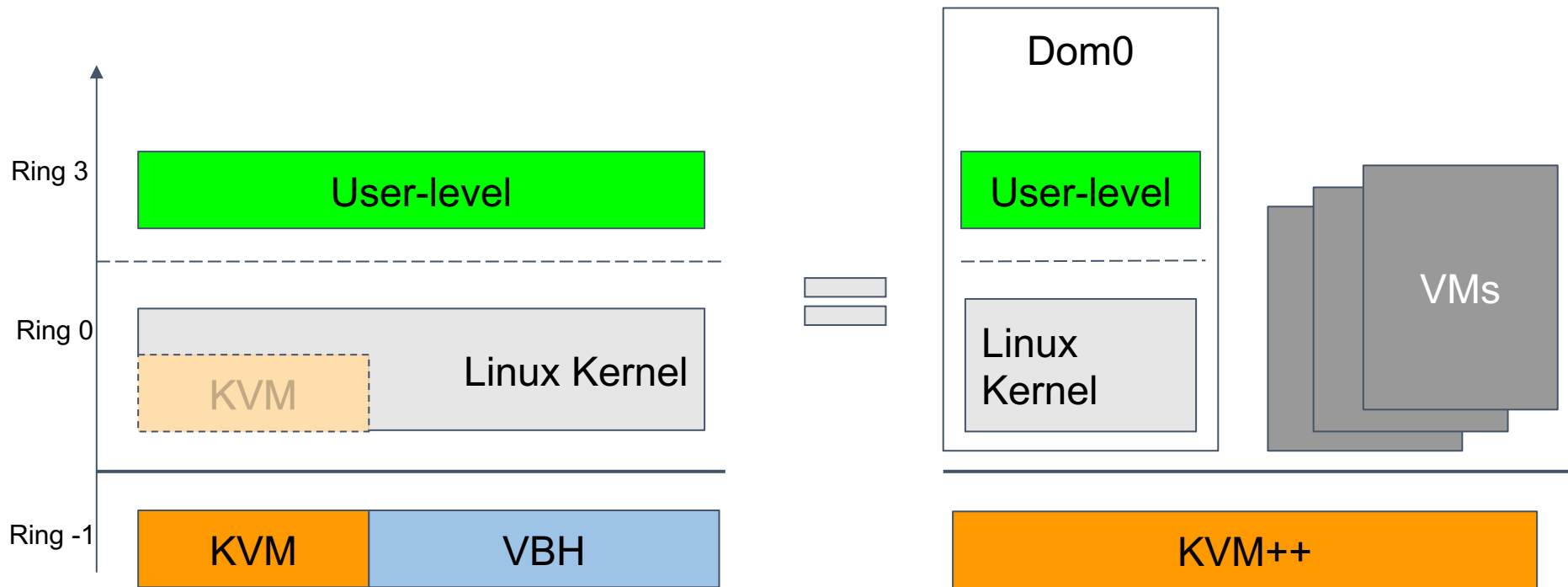
[- Jun Nakajima, Intel & Andrei Lutas, Bitdefender](#)

November 1, 2019

KVM Forum 2019

11

# Making Type-1 Hypervisor From KVM



# Next Steps

- Complete PoC and share the patches
  - Changes to Linux, KVM, etc.
    - Linux guest changes: use existing code for AMD SEV as much as possible
- Propose how to remove guest memory mapping from user-space
  - In Backups
- PoC of “Making Type-1 Hypervisor From KVM”

# Backups

# Remove User-space VMM from TCB

- Deny access to guest register state:
  - Reject ioctls() that get/set guest state
  - Other changes required to hide guest state
- Deny access to guest private memory (example):
  - Add new flag, e.g. VM\_PRIVATE or so, to vm\_flags
  - Kernel removes PTEs for VM\_PRIVATE memory from user-space page tables
  - Guest controls shared vs. private GPAs
  - KVM returns -EFAULT to user-space on private access to memory without VM\_PRIVATE