Linux Plumbers Conference

Richmond, Virginia  |  November 13-15, 2023
Secure AVIC

Securing Interrupt Injection from a Malicious Hypervisor

Authors:

Kishon Vijay Abraham
Suravee Suthikulpanit
Agenda

• Introduction

• Hardware & Software Architecture

• Linux Host / Guest Initialization

• Interrupt Injection Supports
  • IPI interrupts
  • Device interrupts

• Current Status & Issues
Introduction

• Added security for guest APIC registers for SEV-SNP guests

• HW acceleration for performance sensitive APIC register accesses
  • Initially support only Self-IPI and EOI virtualization

• Single vs. Multi-VMPL usage models
  • Currently leverage single-VMPL w/ enlightened guest.
  • #VC handler is responsible for emulating additional functionality

• Only support x2APIC mode via x2APIC MSRs
**HW Architecture**

- **Guest APIC Backing Page**
  - Allocated and managed by Guest
  - Host APIC backing page allocation still exist
    - Cache pending interrupts in IRR

- **AllowedIRR[0..7] Registers (Guest-controlled)**
  - New field to indicate the interrupt vectors which the guest allows the hypervisor to send.

- **RequestedIRR[0..7] Registers (Host-Controlled)**
  - Each set bit in RequestedIRR registers, would be set in the APIC backing page IRR registers by the microcode if the same bits are set in the AllowedIRR registers.
### SW Architecture

- **Guest send IPI**
- **Guest APIC Register Accesses**
- **Guest Updates IRQ vector**
- **Guest APIC Registers**
  - Virtualized by HW
  - Guest Private Memory
  - VMCB
  - #VMGEXIT
  - #VMEXIT
- **AVIC_INCOMPLETE_IPI**
  - #VC Handler
- **AVIC_NOACCEL**
  - #VMEXIT Handler
  - Notify running vCPU
- **Notify running guest vCPU**
- **Guest send IPI**
- **Virtualized device inject interrupts**
- **Emulated device inject interrupts**
- **Requested IRRs**
- **Allowed IRRs**
- **Guest APIC Backing Page**
- **Shadow APIC Backing Page** (SW only)
- **MSR C001_011B**
  - AVIC Doorbell

**VMRUN**

**Virtualization Model**

- Guest Kernel
- Guest #VC Handler
- Guest APIC

**Hardware Virtualization**

- Virtualized by HW
- Notify running guest vCPU
- Notify running guest vCPU
<table>
<thead>
<tr>
<th></th>
<th>Emulated x2APIC</th>
<th>x2AVIC</th>
<th>Secure AVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>APIC Backing Page</strong></td>
<td>Owned by Hypervisor</td>
<td>Owned by Hypervisor</td>
<td>Owned by Guest</td>
</tr>
<tr>
<td><strong>Register Access</strong></td>
<td>All Read and Write: VMEXIT and Handled by hypervisor</td>
<td>• Most Reads are accelerated by HW except extended APIC register</td>
<td>• All accesses to ICR, TPR and EOI registers are accelerated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Most writes results in VMEXIT except ICR, TPR, EOI register</td>
<td>• For other registers, access must be handled by #VC handler</td>
</tr>
<tr>
<td><strong>Self IPI</strong></td>
<td>VMEXIT and injected via event injection</td>
<td>Accelerated by HW</td>
<td>Accelerated by HW</td>
</tr>
<tr>
<td><strong>Broadcast IPI</strong></td>
<td>VMEXIT and injected via event injection</td>
<td>• Accelerated by HW if target vCPU is running</td>
<td>• Self IPI is accelerated by HW</td>
</tr>
<tr>
<td>(All Including Self)</td>
<td></td>
<td>• VMEXIT AVIC_INCOMPLETE_IPI to reschedule vCPU.</td>
<td>• #VC handler for all other target vCPU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VMEXIT to ring doorbell or wakeup target vCPU (by hypervisor)</td>
</tr>
<tr>
<td><strong>Broadcast IPI</strong></td>
<td>VMEXIT and injected via event injection</td>
<td>• Accelerated by HW if target vCPU is running</td>
<td>#VC handler for all target vCPU</td>
</tr>
<tr>
<td>(All Excluding Self)</td>
<td></td>
<td>• VMEXIT AVIC_INCOMPLETE_IPI to reschedule vCPU.</td>
<td>VMEXIT to ring doorbell or wakeup target vCPU (by hypervisor)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Emulated Device</strong></td>
<td>VMEXIT and injected via event injection</td>
<td>• HV inject by updating APIC IRR register</td>
<td>Guest updates AllowedIRRs</td>
</tr>
<tr>
<td>Interrupt Injection</td>
<td></td>
<td>• HV ring doorbell or wakeup target vCPU</td>
<td>HV updates RequestedIRR and UpdateIRR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HV ring doorbell or wakeup target vCPU</td>
</tr>
<tr>
<td><strong>Multiple Pending</strong></td>
<td>VMEXIT for each interrupt using VINTR EXIT</td>
<td>HW invokes ISR for each set IRR</td>
<td>HW invokes ISR for each set bits of [Allowed</td>
</tr>
<tr>
<td><strong>Interrupt</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Initialization: Secure AVIC (Host)

- New `kvm_amd` kernel module option:
  - `kvm_amd.secure_avic`
    (mutual exclusive to the `avic` parameter)

- Checking CPUID for the feature support

- Enabling the feature
  - Set the `SEV_FEATURES[SecureAvic]` bit in the VMSA.

Diagram:

- Start SEV/SNP Guest
- Secure AVIC: Secure Avic == 1
  - kvm_amd module_param
  - CPUID Fn8000_001F_EAX[SecureAvic] == 1
- SNP Flow
  - sev_es_save_area.sev_features
  - SVM_SEV_FEAT_SECURE_AVIC
  - SNP Flow
Initialization: Secure AVIC (Guest)

- Check MSR_C0010131 SEV STATUS [SecureAvicEn] bit
  - Indicates Secure AVIC support for guest

- Allocate Guest APIC Backing Page
  - Program backing page 4k-aligned GPA into the MSR_C001_0138[GuestApicBackingPagePtr]
  - Pinned in while in Guest mode (during vmrun)
  - Page is marked private (encrypted) in RMP table.

- Enabling Secure AVIC feature
  - Set MSR_C0010138[SecureAvicEn] bit
### Guest

- **WRMSR ICR (x2APIC)**
  - **DSH == 3 (All excl self)**
  - **DSH == 2 (All incl self)**
  - **DSH == 1 (Self)**
  - **NO** Non-Automatic Exit
  - **YES** WRMSR SelfIPI

### #VC Handler

- **SVM_EXIT_AVIC_INCOMPLETE_IPI**
  - **YES** Read ICR from EXIT_INFO
  - **NO** Target vCPU == Running

### Hypervisor

- **SVM_EXIT_AVIC_INCOMPLETE_IPI**
  - **YES** Read ICR from Backing Page
  - **NO** Update Backing Page IRR (One or more based on DSH and logical/physical APIC mode)

### Details

- **DSH == 0 (Destination)** (One or more based on DSH and logical/physical APIC mode)
- **DSH == 2 (All incl self)**
- **DSH == 3 (All excl self)**
- **DSH == 0 (Destination)** || **DSH == 2 (All incl self)** || **DSH == 3 (All excl self)**
- **Automatic Exit**
- **VMEXIT Ret**
- **Write AVIC Doorbell to notify target vCPU**
- **Wake up target vCPU**
- **Direction to Hypervisor**

**IPI Injection**
Host Device Interrupt Injection

QEMU Injects Interrupt (Emulated Device / Non-AVIC Pass-through Device)

KVM caches interrupt vector to be injected in APIC IRR of the Shadow APIC backing Page

svm_deliver_interrupt()

vCPU = halt

vCPU exit HLT loop

vmenter Guest

IPI to physical CPU executing vCPU (smp_send_reschedule())

INTR VMEXIT

Copy IRR of shadow APIC Backing Page to RequestedIRR And clear the bit from The shadow APIC Backing Page

Set UpdateIRR bit

Disable PV-EOI

VMRUN

VMEXIT

Clear RequestedIRR and UpdateIRR
Guest Device Interrupt Injection

- arch/x86/kernel/apic/vector.c is modified to update the AllowedIRR registers.

- Microcode sets IRR bit in APIC backing page based on AllowedIRR and RequestedIRR

- Microcode sets ISR bit in APIC backing page (guest) after interrupt handler is invoked

- When Guest Kernel writes to EOI register, microcode clears ISR bit and invokes interrupt handler of the next set IRR
Status

• Phase #1: (Currently work-in-progress)
  • IPI Interrupt Injection support
  • Emulated device interrupt injection support
  • Successfully boot a VM w/ 128 vCPUs

• Phase #2:
  • NMI Emulation
  • LAPIC Timer Emulation
Issues

• Secure AVIC hardware requires PTE entry for guest APIC backing page to always be present in the nested page table
  • There should be no NPF Exception when HW accesses backing page.

• KVM MMU zaps PTE frequently causing NPF resulting in VMEXIT_BUSY and prevents the vcpu to resume.

• Temporary Workaround
  • Invoke kvm_tdp_mmu_map() to populate PTE of backing page in the page table before VMRUN.
  • However, this workaround does not always guarantee the page to be present.
Q & A