author Sean Christopherson <seanjc@google.com> 2021-10-12 17:35:53 -0700
committer Paolo Bonzini <pbonzini@redhat.com> 2021-10-18 14:07:18 -0400
commit f7d8a19f9a056a05c5c509fa65af472a322abfee (patch)
tree 04669ccbc89b4a4fc577971c33d0f54c5c6e83d1
parent baa1e5ca172ce7bf9554070139482dd7ea919528 (diff)
download linux-f7d8a19f9a056a05c5c509fa65af472a322abfee.tar.gz

diff options

context: 3
space: include
mode: unified

Revert "KVM: x86: Open code necessary bits of kvm_lapic_set_base() at vCPU RESET"

```
Revert a change to open code bits of kvm_lapic_set_base() when emulating
APIC RESET to fix an apic_hw_disabled underflow bug due to arch.apic_base
and apic_hw_disabled being unsyncrhonized when the APIC is created.  If
kvm_arch_vcpu_create() fails after creating the APIC, kvm_free_lapic()
will see the initialized-to-zero vcpu->arch.apic_base and decrement
apic_hw_disabled without KVM ever having incremented apic_hw_disabled.

Using kvm_lapic_set_base() in kvm_lapic_reset() is also desirable for a
potential future where KVM supports RESET outside of vCPU creation, in
which case all the side effects of kvm_lapic_set_base() are needed, e.g.
to handle the transition from x2APIC => xAPIC.

Alternatively, KVM could temporarily increment apic_hw_disabled (and call
kvm_lapic_set_base() at RESET), but that's a waste of cycles and would
impact the performance of other vCPUs and VMs.  The other subtle side
effect is that updating the xAPIC ID needs to be done at RESET regardless
of whether the APIC was previously enabled, i.e. kvm_lapic_reset() needs
an explicit call to kvm_apic_set_xapic_id() regardless of whether or not
kvm_lapic_set_base() also performs the update.  That makes stuffing the
enable bit at vCPU creation slightly more palatable, as doing so affects
only the apic_hw_disabled key.

Opportunistically tweak the comment to explicitly call out the connection
between vcpu->arch.apic_base and apic_hw_disabled, and add a comment to
call out the need to always do kvm_apic_set_xapic_id() at RESET.

Underflow scenario:

  kvm_vm_ioctl() {
    kvm_vm_ioctl_create_vcpu() {
      kvm_arch_vcpu_create() {
        if (something_went_wrong)
          goto fail_free_lapic;
        /* vcpu->arch.apic_base is initialized when something_went_wrong is false. */
        kvm_vcpu_reset() {
          kvm_lapic_reset(struct kvm_vcpu *vcpu, bool init_event) {
            vcpu->arch.apic_base = APIC_DEFAULT_PHYS_BASE | MSR_IA32_APICBASE_ENABLE;
          }
        }
        return 0;
      fail_free_lapic:
        kvm_free_lapic() {
          /* vcpu->arch.apic_base is not yet initialized when something_went_wrong is true. */
          if (!(vcpu->arch.apic_base & MSR_IA32_APICBASE_ENABLE))
            static_branch_slow_dec_deferred(&apic_hw_disabled); // <= underflow bug.
        }
        return r;
      }
    }
  }

This (mostly) reverts commit 421221234ada41b4a9f0beeb08e30b07388bd4bd.

Fixes: 421221234ada ("KVM: x86: Open code necessary bits of kvm_lapic_set_base() at vCPU RESET")
Reported-by: syzbot+9fc046ab2b0cf295a063@syzkaller.appspotmail.com
Debugged-by: Tetsuo Handa <penguin-kernel@i-love.sakura.ne.jp>
Signed-off-by: Sean Christopherson <seanjc@google.com>
Message-Id: <20211013003554.47705-2-seanjc@google.com>
Signed-off-by: Paolo Bonzini <pbonzini@redhat.com>
```

Diffstat

-rw-r--r-- arch/x86/kvm/lapic.c 18

1 files changed, 11 insertions, 7 deletions

```
diff --git a/arch/x86/kvm/lapic.c b/arch/x86/kvm/lapic.c
index 76fb009212037..7af25304bda9a 100644
--- a/arch/x86/kvm/lapic.c
+++ b/arch/x86/kvm/lapic.c
@@ -2321,13 +2321,14 @@ EXPORT_SYMBOL_GPL(kvm_apic_update_apicv);
 void kvm_lapic_reset(struct kvm_vcpu *vcpu, bool init_event)
 {
        struct kvm_lapic *apic = vcpu->arch.apic;
+       u64 msr_val;
        int i;

        if (!init_event) {
-               vcpu->arch.apic_base = APIC_DEFAULT_PHYS_BASE |
-                                      MSR_IA32_APICBASE_ENABLE;
+               msr_val = APIC_DEFAULT_PHYS_BASE | MSR_IA32_APICBASE_ENABLE;
                if (kvm_vcpu_is_reset_bsp(vcpu))
-                       vcpu->arch.apic_base |= MSR_IA32_APICBASE_BSP;
+                       msr_val |= MSR_IA32_APICBASE_BSP;
+               kvm_lapic_set_base(vcpu, msr_val);
        }

        if (!apic)
@@ -2336,11 +2337,9 @@ void kvm_lapic_reset(struct kvm_vcpu *vcpu, bool init_event)
        /* Stop the timer in case it's a reset to an active apic */
        hrtimer_cancel(&apic->lapic_timer.timer);

-       if (!init_event) {
-               apic->base_address = APIC_DEFAULT_PHYS_BASE;
-
+       /* The xAPIC ID is set at RESET even if the APIC was already enabled. */
+       if (!init_event)
                kvm_apic_set_xapic_id(apic, vcpu->vcpu_id);
-       }
        kvm_apic_set_version(apic->vcpu);

        for (i = 0; i < KVM_APIC_LVT_NUM; i++)
@@ -2481,6 +2480,11 @@ int kvm_create_lapic(struct kvm_vcpu *vcpu, int timer_advance_ns)
```

```
                lapic_timer_advance_dynamic = false;
        }

+       /*
+        * Stuff the APIC ENABLE bit in lieu of temporarily incrementing
+        * apic_hw_disabled; the full RESET value is set by kvm_lapic_reset().
+        */
+       vcpu->arch.apic_base = MSR_IA32_APICBASE_ENABLE;
        static_branch_inc(&apic_sw_disabled.key); /* sw disabled at reset */
        kvm_iodevice_init(&apic->dev, &apic_mmio_ops);
```