author	Sean Christopherson <seanjc@google.com>	2025-04-01 08:05:04 -0700
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2025-04-20 10:18:22 +0200
commit	f5cbe725b7477b4cd677be1b86b4e08f90572997 (patch)	
tree	a03d8161f20d44f7061420095fd65e7289a242d2	
parent	fc4ba91278e2bee8dabb5e0aac3c150abc07f51b (diff)	
download	linux-f5cbe725b7477b4cd677be1b86b4e08f90572997.tar.gz	

KVM: x86: Acquire SRCU in KVM_GET_MP_STATE to protect guest memory accesses

```
commit ef01cac401f18647d62720cf773d7bb0541827da upstream.

Acquire a lock on kvm->srcu when userspace is getting MP state to handle a
rather extreme edge case where "accepting" APIC events, i.e. processing
pending INIT or SIPI, can trigger accesses to guest memory.  If the vCPU
is in L2 with INIT *and* a TRIPLE_FAULT request pending, then getting MP
state will trigger a nested VM-Exit by way of ->check_nested_events(), and
emuating the nested VM-Exit can access guest memory.

The splat was originally hit by syzkaller on a Google-internal kernel, and
reproduced on an upstream kernel by hacking the triple_fault_event_test
selftest to stuff a pending INIT, store an MSR on VM-Exit (to generate a
memory access on VMX), and do vcpu_mp_state_get() to trigger the scenario.

  ============================
  WARNING: suspicious RCU usage
  6.14.0-rc3-b112d356288b-vmx/pi_lockdep_false_pos-lock #3 Not tainted
  -----------------------------
  include/linux/kvm_host.h:1058 suspicious rcu_dereference_check() usage!

  other info that might help us debug this:

  rcu_scheduler_active = 2, debug_locks = 1
  1 lock held by triple_fault_ev/1256:
   #0: ffff88810df5a330 (&vcpu->mutex){+.+.}-{4:4}, at: kvm_vcpu_ioctl+0x8b/0x9a0 [kvm]

  stack backtrace:
  CPU: 11 UID: 1000 PID: 1256 Comm: triple_fault_ev Not tainted 6.14.0-rc3-b112d356288b-vmx #3
  Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 0.0.0 02/06/2015
  Call Trace:
   <TASK>
   dump_stack_lvl+0x7f/0x90
   lockdep_rcu_suspicious+0x144/0x190
   kvm_vcpu_gfn_to_memslot+0x156/0x180 [kvm]
   kvm_vcpu_read_guest+0x3e/0x90 [kvm]
   read_and_check_msr_entry+0x2e/0x180 [kvm_intel]
   __nested_vmx_vmexit+0x550/0xde0 [kvm_intel]
   kvm_check_nested_events+0x1b/0x30 [kvm]
   kvm_apic_accept_events+0x33/0x100 [kvm]
   kvm_arch_vcpu_ioctl_get_mpstate+0x30/0x1d0 [kvm]
   kvm_vcpu_ioctl+0x33e/0x9a0 [kvm]
```

```
   __x64_sys_ioctl+0x8b/0xb0
   do_syscall_64+0x6c/0x170
   entry_SYSCALL_64_after_hwframe+0x4b/0x53
   </TASK>

Cc: stable@vger.kernel.org
Signed-off-by: Sean Christopherson <seanjc@google.com>
Message-ID: <20250401150504.829812-1-seanjc@google.com>
Signed-off-by: Paolo Bonzini <pbonzini@redhat.com>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

Diffstat

-rw-r--r-- arch/x86/kvm/x86.c 4

1 files changed, 4 insertions, 0 deletions

```
diff --git a/arch/x86/kvm/x86.c b/arch/x86/kvm/x86.c
index 50aaed9830cc63..0d0ef9990814b8 100644
--- a/arch/x86/kvm/x86.c
+++ b/arch/x86/kvm/x86.c
@@ -11765,6 +11765,8 @@ int kvm_arch_vcpu_ioctl_get_mpstate(struct kvm_vcpu *vcpu,
        if (kvm_mpx_supported())
                kvm_load_guest_fpu(vcpu);

+       kvm_vcpu_srcu_read_lock(vcpu);
+
        r = kvm_apic_accept_events(vcpu);
        if (r < 0)
                goto out;
@@ -11778,6 +11780,8 @@ int kvm_arch_vcpu_ioctl_get_mpstate(struct kvm_vcpu *vcpu,
                mp_state->mp_state = vcpu->arch.mp_state;

 out:
+       kvm_vcpu_srcu_read_unlock(vcpu);
+
        if (kvm_mpx_supported())
                kvm_put_guest_fpu(vcpu);
        vcpu_put(vcpu);
```