



author Sean Christopherson <seanjc@google.com> 2022-11-30 23:08:58 +0000
committer Greg Kroah-Hartman <gregkh@linuxfoundation.org> 2023-03-17 08:50:17 +0100
commit [e136e969d268b9b89329c816c002e53f60e82985](#) (patch)
tree [782db444dcf6b8a7c04a61a887a8bcf8045f62fb](#)
parent [adc0dd8b043932af005f03028592f725116b6606](#) (diff)
download [linux-e136e969d268b9b89329c816c002e53f60e82985.tar.gz](#)

diff options

context: ▼
space: ▼
mode: ▼

KVM: VMX: Do `_all_` initialization before exposing `/dev/kvm` to userspace

[Upstream commit [e32b120071ea114efc0b4ddd439547750b85f618](#)]

Call `kvm_init()` only after `_all_` setup is complete, as `kvm_init()` exposes `/dev/kvm` to userspace and thus allows userspace to create VMs (and call other `ioctl`s). E.g. KVM will encounter a NULL pointer when attempting to add a vCPU to the per-CPU `loaded_vmcss_on_cpu` list if userspace is able to create a VM before `vmx_init()` configures said list.

BUG: kernel NULL pointer dereference, address: 0000000000000008

#PF: supervisor write access in kernel mode

#PF: error_code(0x0002) - not-present page

PGD 0 P4D 0

Oops: 0002 [#1] SMP

CPU: 6 PID: 1143 Comm: stable Not tainted 6.0.0-rc7+ #988

Hardware name: QEMU Standard PC (Q35 + ICH9, 2009), BIOS 0.0.0 02/06/2015

RIP: 0010:vmx_vcpu_load_vmcs+0x68/0x230 [kvm_intel]

<TASK>

vmx_vcpu_load+0x16/0x60 [kvm_intel]

kvm_arch_vcpu_load+0x32/0x1f0 [kvm]

vcpu_load+0x2f/0x40 [kvm]

kvm_arch_vcpu_create+0x231/0x310 [kvm]

kvm_vm_ioctl+0x79f/0xe10 [kvm]

? handle_mm_fault+0xb1/0x220

__x64_sys_ioctl+0x80/0xb0

do_syscall_64+0x2b/0x50

entry_SYSCALL_64_after_hwframe+0x46/0xb0

RIP: 0033:0x7f5a6b05743b

</TASK>

Modules linked in: vhost_net vhost vhost_iotlb tap kvm_intel(+) kvm irqbypass

Cc: stable@vger.kernel.org

Signed-off-by: Sean Christopherson <seanjc@google.com>

Message-Id: <20221130230934.1014142-15-seanjc@google.com>

Signed-off-by: Paolo Bonzini <pbonzini@redhat.com>

Signed-off-by: Sasha Levin <sashal@kernel.org>

Diffstat

-rw-r--r-- [arch/x86/kvm/vmx/vmx.c](#) 30

1 files changed, 19 insertions, 11 deletions

```

diff --git a/arch/x86/kvm/vmx/vmx.c b/arch/x86/kvm/vmx/vmx.c
index 81a23ae4f872b8..bc868958e91fe8 100644
--- a/arch/x86/kvm/vmx/vmx.c
+++ b/arch/x86/kvm/vmx/vmx.c
@@ -8456,19 +8456,23 @@ static void vmx_cleanup_l1d_flush(void)
    l1tf_vmx_mitigation = VMENTER_L1D_FLUSH_AUTO;
}

-static void vmx_exit(void)
+static void __vmx_exit(void)
{
+    allow_smaller_maxphyaddr = false;
+
#ifdef CONFIG_KEXEC_CORE
    RCU_INIT_POINTER(crash_vmclean_loaded_vmcss, NULL);
    synchronize_rcu();
#endif
+    vmx_cleanup_l1d_flush();
+}

+static void vmx_exit(void)
+{
    kvm_exit();
    kvm_x86_vendor_exit();

-    vmx_cleanup_l1d_flush();
-
-    allow_smaller_maxphyaddr = false;
+    __vmx_exit();
}
module_exit(vmx_exit);

@@ -8513,11 +8517,6 @@ static int __init vmx_init(void)
    if (r)
        return r;

-    r = kvm_init(&vmx_init_ops, sizeof(struct vcpu_vmx),
-                __alignof__(struct vcpu_vmx), THIS_MODULE);
-    if (r)
-        goto err_kvm_init;
-
    /*
     * Must be called after common x86 init so enable_ept is properly set
     * up. Hand the parameter mitigation value in which was stored in
@@ -8551,11 +8550,20 @@ static int __init vmx_init(void)
    if (!enable_ept)
        allow_smaller_maxphyaddr = true;

+    /*
+     * Common KVM initialization _must_ come last, after this, /dev/kvm is
+     * exposed to userspace!
+     */
+    r = kvm_init(&vmx_init_ops, sizeof(struct vcpu_vmx),
+                __alignof__(struct vcpu_vmx), THIS_MODULE);
+    if (r)
+        goto err_kvm_init;
+
    return 0;

-err_l1d_flush:
-    vmx_exit();

```

```
err_kvm_init:
+     __vmx_exit();
+err_lld_flush:
    kvm_x86_vendor_exit();
    return r;
}
```