Google

TDX Selftests

Sagi Shahar <sagis@google.com>, Sep 13, 2022

TDX development streams

- TDX capable CPU
- Bios and TDX Module
- TDX Host Kernel
- UEFI
- TDX Guest Kernel
- Userspace VMM

How are we using selftests

- We've been using TDX selftests for ~year
- Sent first RFC upstream July last year
- Sent RFC V2 last month
- Recently shared our selftests development on GitHub
  - https://github.com/googleprodkernel/linux-cc/tree/selftests
- Used internally for
  - Platform compatibility test
  - Reproducers and edge cases
  - Spec conformance tests
  - New feature development

- Platform compatibility test
- Spec conformance tests
- Reproducers and edge cases
- New feature development

Platform compatibility

- TDX capable CPU
- Bios and TDX Module
- TDX Host Kernel
- UEFI
- TDX Guest Kernel
- Userspace VMM

- Platform compatibility test
- Spec conformance tests
- Reproducers and edge cases
- New feature development

Spec conformance tests

- GHCI protocol
  - CPUID
  - IO
  - MMIO
  - MSR access

MSR access bug

[RFC V5] KVM: TDX: Handle TDX PV rdmsr hypercall (March 4, 2022)

```
+static int tdx_emulate_rdmsr(struct kvm_vcpu *vcpu)
+{
+    u32 index = tdvmcall_p1_read(vcpu);
+    u64 data;
+
+    if (kvm_get_msr(vcpu, index, &data)) {
+        trace_kvm_msr_read_ex(index);
+        tdvmcall_set_return_code(vcpu, TDG_VP_VMCALL_INVALID_OPERAND);
+        return 1;
+    }
+    trace_kvm_msr_read(index, data);
+
+    tdvmcall_set_return_code(vcpu, TDG_VP_VMCALL_SUCCESS);
+    tdvmcall_set_return_val(vcpu, data);
+    return 1;
+}
```

KVM: x86: Only do MSR filtering when access MSR by rdmsr/wrmsr (March 7, 2022)

```
@@ -1820,9 +1817,6 @@ int __kvm_get_msr(struct kvm_vcpu *vcpu, u32 index, u64 *data,
     struct msr_data msr;
     int ret;

-    if (!host_initiated && !kvm_msr_allowed(vcpu, index, KVM_MSR_FILTER_READ))
-        return KVM_MSR_RET_FILTERED;

     switch (index) {
     case MSR_TSC_AUX:
         if (!kvm_is_supported_user_return_msr(MSR_TSC_AUX))
@@ -1859,6 +1853,20 @@ static int kvm_get_msr_ignored_check(struct kvm_vcpu *vcpu,
     return ret;
 }

+static int kvm_get_msr_with_filter(struct kvm_vcpu *vcpu, u32 index, u64 *data)
+{
+    if (!kvm_msr_allowed(vcpu, index, KVM_MSR_FILTER_READ))
+        return KVM_MSR_RET_FILTERED;
+    return kvm_get_msr_ignored_check(vcpu, index, data, false);
+}
```

[RFC V6] KVM: TDX: Handle TDX PV rdmsr/wrmsr hypercall (May 5, 2022)

```
     +static int tdx_emulate_rdmsr(struct kvm_vcpu *vcpu)
+{
+    u32 index = tdvmcall_a0_read(vcpu);
+    u64 data;
+
+    if (kvm get msr(vcpu, index, &data)) {
+        trace kvm msr read ex(index);
+        tdvmcall set_return_code(vcpu, TDG_VP_VMCALL_INVALID_OPERAND);
+        return 1;
+    }
+    trace_kvm_msr_read(index, data);
+
+    tdvmcall set return code(vcpu, TDG VP_VMCALL_SUCCESS);
+    tdvmcall_set_return_val(vcpu, data);
+    return 1;
+}
```

MSR access bug

- Not caught by developer
- Not caught by code review
- Not caught by running VM
- Could have led to security violation

- Platform compatibility test
- Spec conformance tests
- Reproducers and edge cases
- New feature development

Reproducers and edge cases

- Host accessing guest shared memory
- Host accessing guest private memory
- Host writing into guest secure EPT tables

- Platform compatibility test
- Spec conformance tests
- Reproducers and edge cases
- New feature development

New feature development

- Some features require changing multiple components (back to house of cards)
- One example is copyless migration
  - Requires both KVM and VMM changes (and possibly more)
  - Hard to Debug when dealing with a full VM
    - Complicated memory access
    - Indeterministic behavior
    - Need to support full feature for initial test

New feature development – Divide and conquer

- Develop different components separately
- Incrementally support more complicated use cases
  - VM with no memory
  - VM with only private memory
  - VM with both private and shared memory
  - etc.

Selftests overview

```
TDX_GUEST_FUNCTION(guest_dummy_exit) {
        tdvmcall_success();
}

void verify_td_lifecycle(void) {
        struct kvm_vcpu *vcpu;
        struct kvm_vm *vm;

        vm = vm_create_tdx();
        initialize_td(vm);

        vcpu = vm_vcpu_add_tdx(vm, 0);

        /* Setup and initialize VM memory */
        prepare_source_image(vm, guest_dummy_exit,
                             TDX_FUNCTION_SIZE(guest_dummy_exit), 0);
        finalize_td_memory(vm);

        vcpu_run(vcpu);
        CHECK_GUEST_COMPLETION(vcpu);

        kvm_vm_free(vm);
}
```

Selftests overview

- Run as a user space application
- Both host and guest code can be written together
- Uses GHCI to communicate between guest and host

Selftests overview

```
TDX_GUEST_FUNCTION(guest_dummy_exit) {
        tdvmcall_success();
}

void verify_td_lifecycle(void) {
        struct kvm_vcpu *vcpu;
        struct kvm_vm *vm;

        vm = vm_create_tdx();
        initialize_td(vm);

        vcpu = vm_vcpu_add_tdx(vm, 0);

        /* Setup and initialize VM memory */
        prepare_source_image(vm, guest_dummy_exit,
                TDX_FUNCTION_SIZE(guest_dummy_exit), 0);
        finalize_td_memory(vm);

        vcpu_run(vcpu);
        CHECK_GUEST_COMPLETION(vcpu);

        kvm_vm_free(vm);
}
```

```
inline void tdvmcall_success(void)
{
        uint64_t code = 0;

        tdvmcall_io(TDX_SUCCESS_PORT, /*size=*/4,
                    TDX_IO_WRITE, &code);
}

inline uint64_t tdvmcall_io(uint64_t port, uint64_t size,
                            uint64_t *data)
{
        struct kvm_regs regs;

        memset(&regs, 0, sizeof(regs));
        regs.r11 = TDX_INSTRUCTION_IO;
        regs.r12 = size;
        regs.r13 = write;
        regs.r14 = port;
        regs.r15 = *data;
        regs.rcx = 0xFC00;
        tdcall(&regs);
        return regs.r10;
}

#define CHECK_GUEST_COMPLETION(VCPU)   \
        (TEST_ASSERT(                  \
            ((VCPU)->run->exit_reason == KVM_EXIT_IO) &&   \
            ((VCPU)->run->io.port == TDX_SUCCESS_PORT) && \
            ((VCPU)->run->io.size == 4) &&                 \
            ((VCPU)->run->io.direction == TDX_IO_WRITE)))
```

Summary

Selftests are useful

- Simple to write and run
- Easy to test corner cases and negative tests
- Doesn’t require full system setup for testing

We would appreciate community support

- RFC V2
- https://github.com/googleprodkernel/linux-cc/tree/selftests