# x86 : Virtualization

Gabriel Laskar <gabriel@lse.epita.fr>

LSE Security System

# Basics : What is it ?

- Virtual Machine
- Hypervisor
- Virtual machine monitor

LSE Security System

# Basics : Virtualization vs Emulation

- CPU Emulation : Interpret code in order to execute the same behavior
- CPU Virtualization : Execute on real hardware, but in a controlled way

# VM Requirements

*« For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of privileged instructions »*

-- Popek & Goldberg

LSE Security System

# Virtualization Solutions

- Xen
- Qemu/KVM
- VMWare ESX
- VMWare Workstation
- VirtualBox

# Other Kind of Virtualization

- Paravirtualization
- Containers

LSE Security System

# CPU Virtualization

- Run the VMM at a higher level of privilege
- Sensitive instructions will trap and the VMM will emulate them.

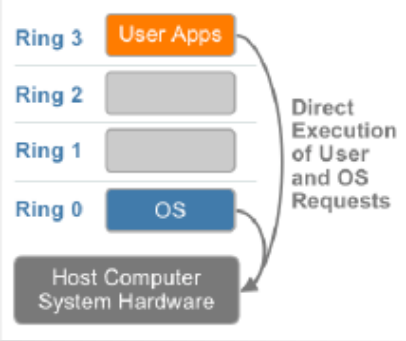
LSE Security System

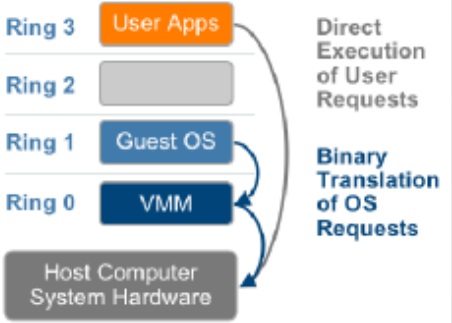
# Virtualize the “unvirtualizable”

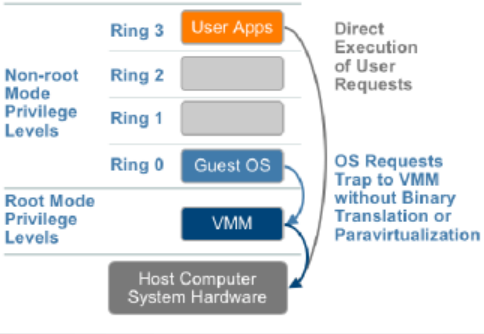
- Binary Rewriting
- Para-virtualization
- HVM

LSE Security System

# Rings & Virtualization




LSE Security System

# vt-x

- root vs non-root mode
- VMCS
- Instructions
- What can trap ?

# vt-x : instructions

- vmptrld, vmptrst
- vmclear
- vmread, vmwrite
- vmlaunch, vmresume
- vmxoff, vmxon
- invept, invvpid
- vmcall, vmfunc

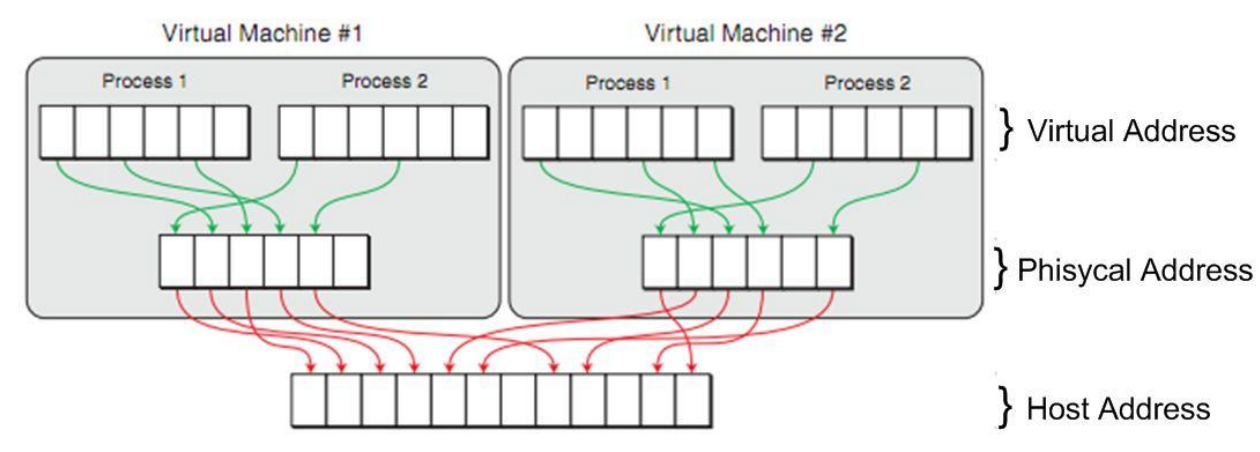
LSE Security System

# EPT

- No Shadow Page Tables
- A second translation Layer
- translation : physical → guest-physical

# Memory Virtualization

- shadow page tables
- EPT



# Example Hypervisor: Qemu/KVM

- KVM is the Linux Hypervisor
- Splitted in 2 parts :
  - kvm : kernel module
  - qemu : device emulation, vm setup

LSE Security System

# KVM

- Leverage Linux APIs & subsystems for Virtualization
- 3 modules : kvm.ko, kvm-intel.ko, kvm-amd.ko
- code size :
  - ~7kloc arch-independant code
  - ~33kloc arch-dependant code for x86 (~8k for arm)

LSE Security System

# KVM

- Expose virtualization api to the userland
- Use only Hardware virtualization instructions
- small size
- reuse linux apis when possible (scheduling, memory management, events, ...)

# Qemu

- Use also in Xen
- VM creation
- Device emulation

# KVM Api

- VM creation
- Memory assignation
- irq chip
- launch a cpu
- devices

# /dev/kvm

- /dev/kvm expose an anonymous virtual filesystem for the hypervisor
- Every resources are managed through a fd :
  - kvm configuration
  - vm management
  - vcpu management

# /dev/kvm : system fd

- `ioctl(fd, KVM_CREATE_VM)`
- `ioctl(fd, KVM_GET_VCPU_MMAP_SIZE)`
- `ioctl(fd, KVM_GET_MSR_INDEX_LIST)`
- `ioctl(fd, KVM_CHECK_EXTENSION)`

LSE Security System

# kvm extensions

- Multiple architectures and capabilities (and kvm versions)
- Extension system, in order to know what is available
- Around 115 extension (as of 4.1)
- Check for extension before use:
  - `ioctl(kvm_fd, KVM_CHECK_EXTENSION, KVM_CAP_IRQCHIP);`

# Example : vm creation

```
int fd_kvm = open("/dev/kvm", O_RDWR);
int kvm_run_size = ioctl(fd_kvm, KVM_GET_VCPU_MMAP_SIZE,
0);

int fd_vm = ioctl(fd_kvm, KVM_CREATE_VM, 0);

// add space for v8086 TSS (3 pages)
ioctl(fd_vm, KVM_SET_TSS_ADDR, 0xfffffd000);
// add space for identity map for vcpu real mode
ioctl(fd_vm, KVM_SET_IDENTITY_MAP_ADDR, 0xfffffc000);

ioctl(fd_vm, KVM_CREATE_IRQCHIP, 0);
```

LSE Security System

# /dev/kvm : vm fd

- KVM_CREATE_VCPU
- KVM_SET_USER_MEMORY_REGION
- KVM_CREATE_IRQCHIP (extension)
- KVM_{GET,SET}_DEBUGREGS
- KVM_GET_DIRTY_LOG

# Example : Memory Assignation

```
// set memory region
void *addr = mmap(NULL, 10 * MB, PROT_READ | PROT_WRITE,
                  MAP_ANONYMOUS | MAP_PRIVATE, -1, 0);

struct kvm_userspace_memory_region region = {
        .slot = 0,
        .flags = 0,
        .guest_phys_addr = 0x100000,
        .memory_size = 10 * MB,
        .userspace_addr = (__u64)addr
};

ioctl(fd_vm, KVM_SET_USER_MEMORY_REGION, &region);
```

LSE Security System

# /dev/kvm : VCPU fd

- KVM_RUN
- KVM_{GET,SET}_REGS
- KVM_{GET,SET}_SREGS
- KVM_TRANSLATE
- KVM_INTERRUPT (without local apic)
- KVM_{GET,SET}_MSRS
- KVM_SET_CPUID

# Example : VCPU Creation & setup

```
int fd_vcpu = ioctl(fd_vm, KVM_CREATE_VCPU, 0);

struct kvm_sregs sregs;
ioctl(fd_vcpu, KVM_GET_SREGS, &sregs);

#define set_segment(Seg, Base, Limit, G) \
  do {                                   \
    Seg.base = Base;                     \
    Seg.limit = Limit;                   \
    Seg.g = G;                           \
  } while (0)

set_segment(sregs.cs, 0x0, 0xffffffff, 1);
set_segment(sregs.ds, 0x0, 0xffffffff, 1);
set_segment(sregs.ss, 0x0, 0xffffffff, 1);

sregs.cs.db = 1;
sregs.ss.db = 1;
```

```
sregs.cr0 |= 0x01;

ioctl(fd_vcpu, KVM_SET_SREGS, &sregs);

struct kvm_regs regs;
ioctl(fd_vcpu, KVM_GET_REGS, &regs);
regs.rflags = 0x02;
regs.rip = 0x00100f00;
ioctl(fd_vcpu, KVM_SET_REGS, &regs);
```

# Example : Run VM

```
struct kvm_run *run_state =
    mmap(0, kvm_run_size, PROT_READ|PROT_WRITE,
MAP_PRIVATE,
         fd_vcpu, 0);

for (;;) {
        int res = ioctl(fd_vcpu, KVM_RUN, 0);

        switch (run_state->exit_reason) {
                /* … */

        }
}
```

# Exit Reasons

- KVM_EXIT_EXCEPTION
- KVM_EXIT_IO
- KVM_EXIT_MMIO
- KVM_EXIT_SHUTDOWN
- …

LSE Security System

# Port IO

```
case KVM_EXIT_IO:
        if (run_state->io.port == CONSOLE_PORT
           && run_state->io.direction == KVM_EXIT_IO_OUT)
{

                __u64 offset = run_state->io.data_offset;
                __u32 size = run_state->io.size;

                write(STDOUT_FILENO,
                        (char*)run_state + offset, size);
        }
        break;
```

# More? Where is the documentation?

- linux source code:
  - include/uapi/linux/kvm.h
  - Documentation/virtual/kvm/api.txt
  - virt/kvm/
  - arch/x86/kvm/
  - arch/arm/kvm/
- qemu source code
- kvmtool:
  - https://raw.githubusercontent.com/penberg/linux-kvm/master/tools/kvm/README
- As usual Intel® 64 and IA-32 Architectures Software Developer Manuals

LSE Security System