Virtualizing Security

flyyy

Blue-Whale

1 Definitions

Some definitions before begin

2 Virtualizing development and types

3 Virtualizing realization

Qemu-kvm anylize

4 Security

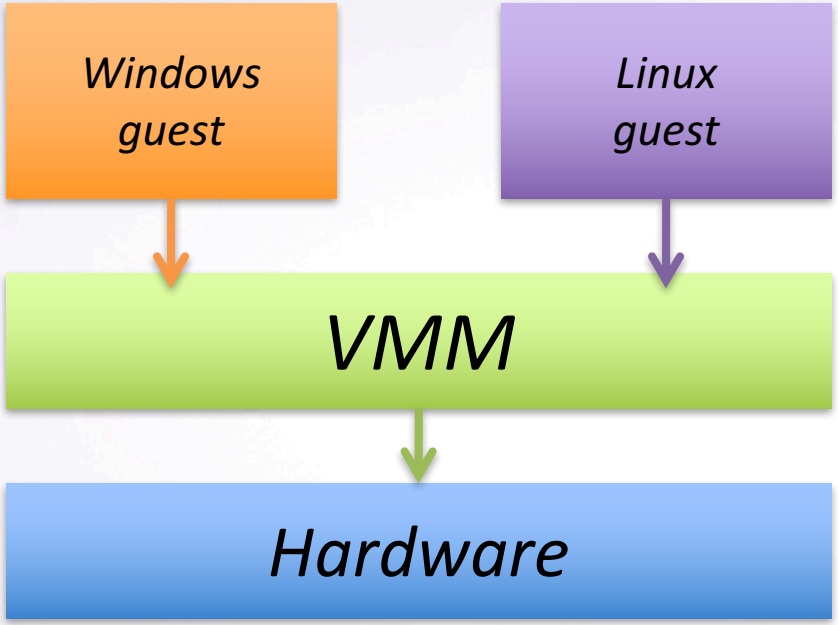
How to escape from virtualizing

Definitions

1. VMM(Vitural Machine Monitor)and hypervisor
2. Host OS and Guest OS
3. Full-virtualization and para-virtualization

   Just talk about full-virtualization.

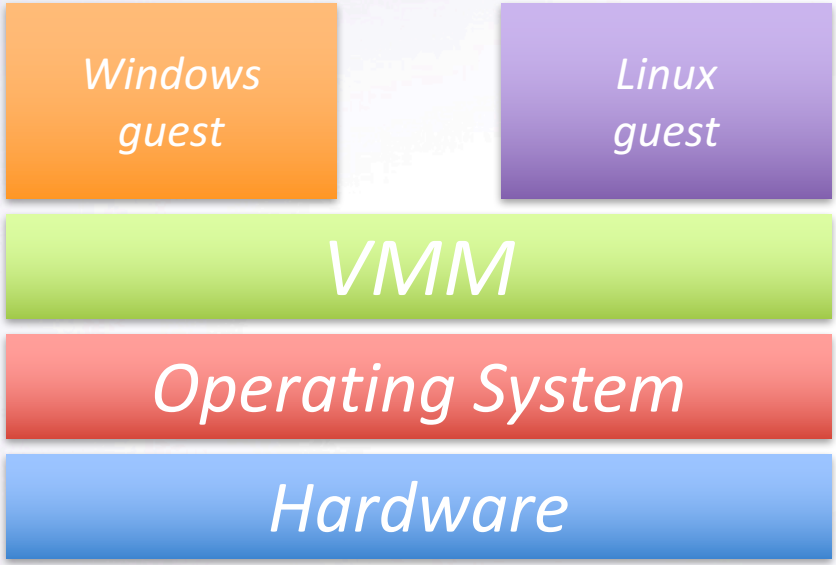
Types



Type-I (Native)

The VMM runs directly on the hardware. It can control the hardware resources.
Examples:Vmware ESXi, Microsoft Hyper-V.

Types



Type-II (Hosted)

The VMM runs as an application.The host OS control the hardware resources.
Examples:VMware workstation ,Qemu

Development

There used to be two ways to realize virtualization:

1. Emulation

   Slow,but can realize virtualization on different architectures.

   i.e.You can run mips ELF by Qemu on x86 architecture

2. Direct native execution

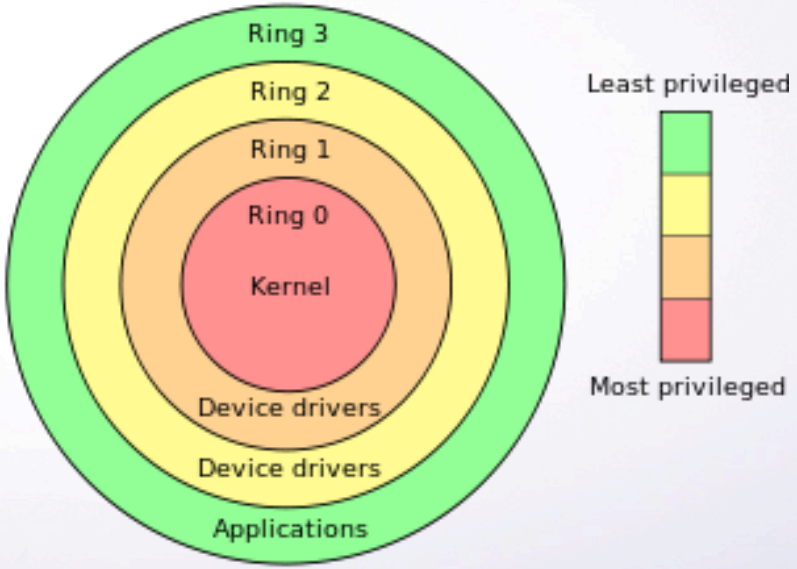
   Fast,but there’ s a problem.

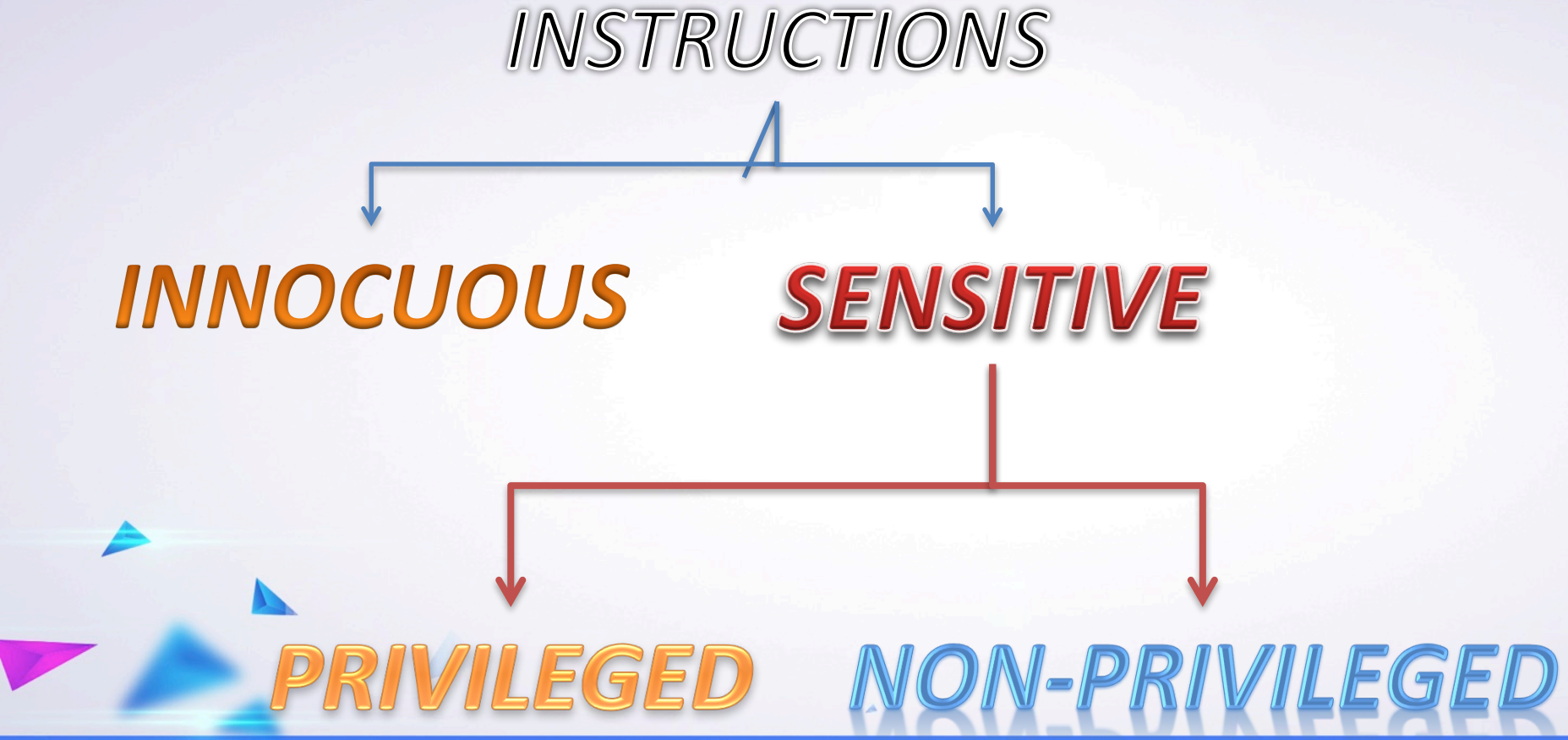
Ring0-Ring3

OS runs in Ring0.

User application runs in ring3.

Linux and windows only use these two levels.




INSTRUCTIONS
INNOCUOUS
SENSITIVE
PRIVILEGED
NON-PRIVILEGED

Sensitive instruction

It can effect system resources or behavior.

Run privileged instruction on ring3 will rise a exception.

We can catch the exception and emulate the privileged instruction.

But not all the sensitive instructions are privileged.Like popfd.

So we need to translate sensitive but not privileged instructions.

Hardware Support

Intel Virtualization Technology

New instruction : Intel Virtual Machine eXtensions (VMS)

AMD Virtualization

New instruction : AMD Secure Virtual Machine （ SVM ）

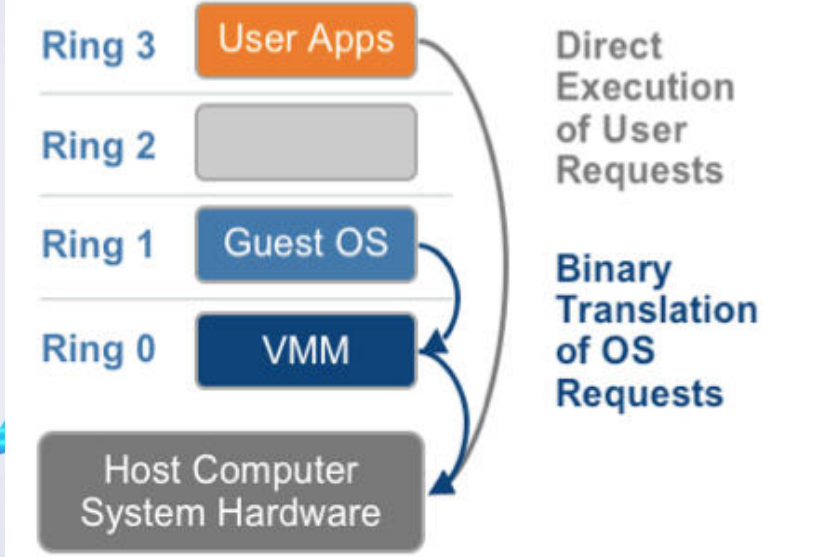
Hardware Support

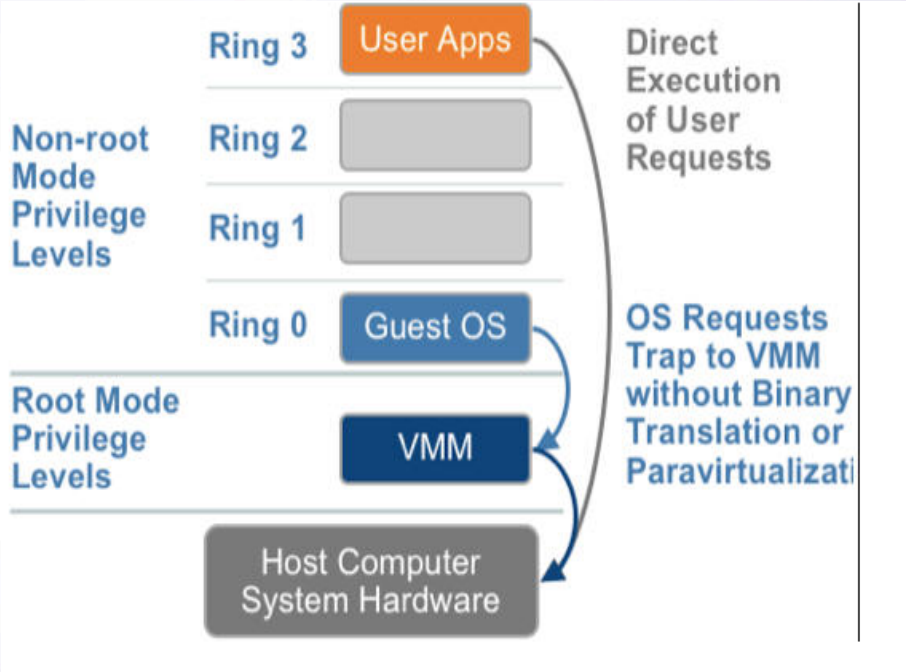
VMX Root operation and VMX non-root operation.

Both mode support Ring0-Ring4

Someone says VMM runs in Ring -1

Changes


Ring 3
User Apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host Computer System Hardware
Direct Execution of User Requests
Binary Translation of OS Requests


Ring 3
User Apps
Non-root Mode Privilege Levels
Ring 2
Ring 1
Ring 0
Guest OS
Root Mode Privilege Levels
VMM
Host Computer System Hardware
Direct Execution of User Requests
OS Requests Trap to VMM without Binary Translation or Paravirtualizati

KVM

Kernel-based Virtual Machine (KVM) is a model of Linux kernel

Provide CPU virtualizing and memory virtualizing

Provide catching of guest OS’ s IO

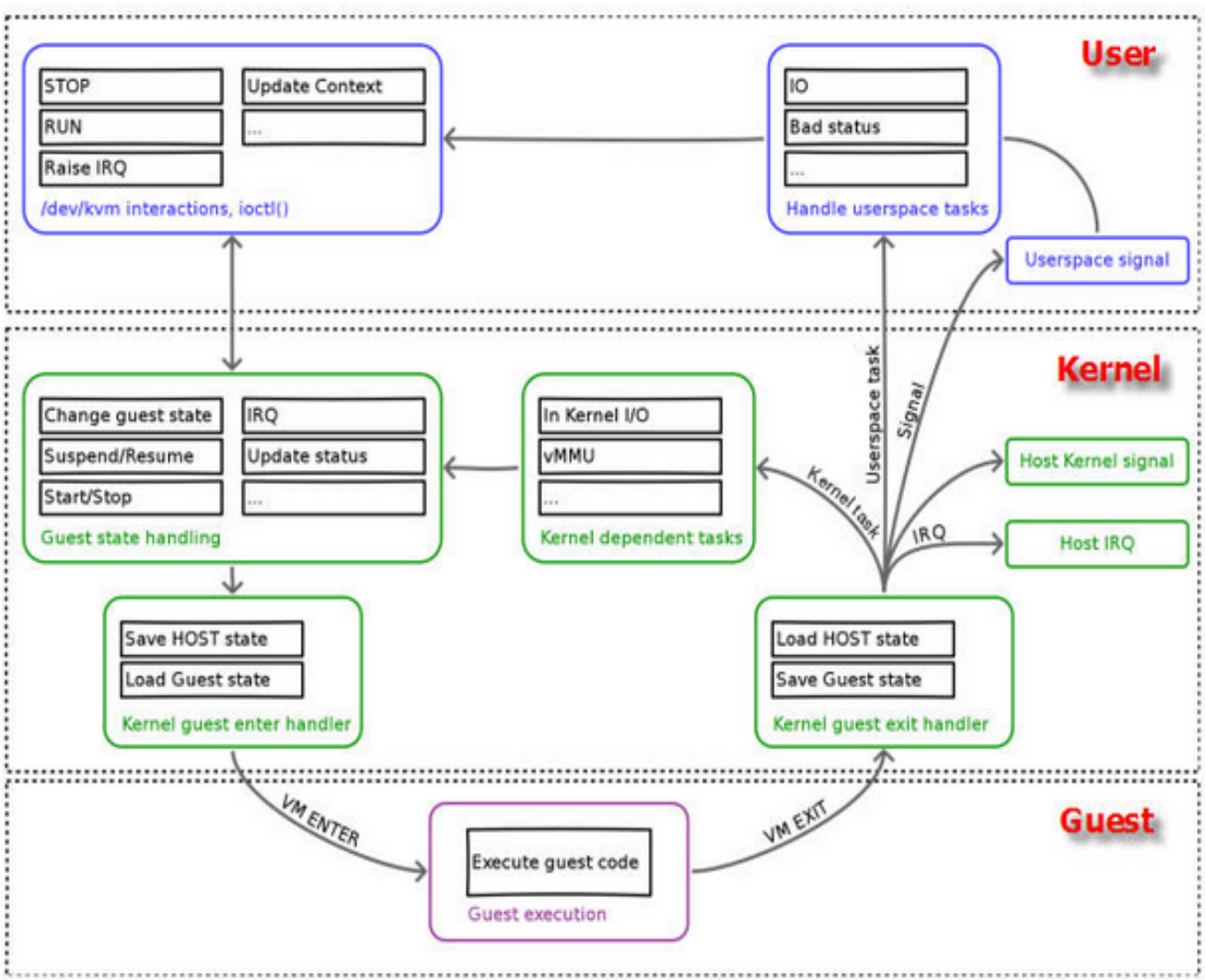
Provide API : /dev/kvm

Qemu

Qemu used to be a emulation sofaware

Now it’ s used with kvm

Provided emulative hardware


User
STOP
Update Context
RUN
...
Raise IRQ
/dev/kvm interactions, ioctl()
IO
Bad status
...
Handle userspace tasks
Userspace signal
Kernel
Change guest state
IRQ
Suspend/Resume
Update status
Start/Stop
...
Guest state handling
In Kernel I/O
vMMU
...
Kernel dependent tasks
Userspace task
Signal
Kernel task
Host Kernel signal
IRQ
Host IRQ
Save HOST state
Load Guest state
Kernel guest enter handler
Load HOST state
Save Guest state
Kernel guest exit handler
Guest
VM ENTER
Execute guest code
Guest execution
VM EXIT

```
static int (*const kvm_vmx_exit_handlers[])(struct kvm_vcpu *vcpu) = {
    [EXIT_REASON_EXCEPTION_NMI]           = handle_exception,
    [EXIT_REASON_EXTERNAL_INTERRUPT]      = handle_external_interrupt,
    [EXIT_REASON_TRIPLE_FAULT]            = handle_triple_fault,
    [EXIT_REASON_NMI_WINDOW]          = handle_nmi_window,
    [EXIT_REASON_IO_INSTRUCTION]          = handle_io,
    [EXIT_REASON_CR_ACCESS]               = handle_cr,
    [EXIT_REASON_DR_ACCESS]               = handle_dr,
    [EXIT_REASON_CPUID]                   = handle_cpuid,
    [EXIT_REASON_MSR_READ]                = handle_rdmsr,
    [EXIT_REASON_MSR_WRITE]               = handle_wrmsr,
    [EXIT_REASON_PENDING_INTERRUPT]       = handle_interrupt_window,
    [EXIT_REASON_HLT]                     = handle_halt,
    [EXIT_REASON_INVD]              = handle_invd,
    [EXIT_REASON_INVLPG]              = handle_invlpg,
    [EXIT_REASON_RDPMC]                   = handle_rdpmc,
    [EXIT_REASON_VMCALL]                  = handle_vmcall,
    [EXIT_REASON_VMCLEAR]                 = handle_vmclear,
    [EXIT_REASON_VMLAUNCH]                = handle_vmlaunch,
    [EXIT_REASON_VMPTRLD]                 = handle_vmptrld,
    [EXIT_REASON_VMPTRST]                 = handle_vmptrst,
    [EXIT_REASON_VMREAD]                  = handle_vmread,
    [EXIT_REASON_VMRESUME]                = handle_vmresume,
    [EXIT_REASON_VMWRITE]                 = handle_vmwrite,
    [EXIT_REASON_VMOFF]                   = handle_vmoff,
    [EXIT_REASON_VMON]                    = handle_vmon,
    [EXIT_REASON_TPR_BELOW_THRESHOLD]     = handle_tpr_below_threshold,
    [EXIT_REASON_APIC_ACCESS]             = handle_apic_access,
    [EXIT_REASON_APIC_WRITE]              = handle_apic_write,
    [EXIT_REASON_EOI_INDUCED]             = handle_apic_eoi_induced,
    [EXIT_REASON_WBINVD]                  = handle_wbinvd,
    [EXIT_REASON_XSETBV]                  = handle_xsetbv,
    [EXIT_REASON_TASK_SWITCH]             = handle_task_switch,
    [EXIT_REASON_MCE_DURING_VMENTRY]      = handle_machine_check,
    [EXIT_REASON_EPT_VIOLATION]       = handle_ept_violation,
    [EXIT_REASON_EPT_MISCONFIG]           = handle_ept_misconfig,
    [EXIT_REASON_PAUSE_INSTRUCTION]       = handle_pause,
    [EXIT_REASON_MWAIT_INSTRUCTION]       = handle_mwait,
    [EXIT_REASON_MONITOR_TRAP_FLAG]       = handle_monitor_trap,
    [EXIT_REASON_MONITOR_INSTRUCTION]     = handle_monitor,
    [EXIT_REASON_INVEPT]                  = handle_invept,
    [EXIT_REASON_INVVPID]                 = handle_invvpid,
    [EXIT_REASON_XSAVES]                  = handle_xsaves,
    [EXIT_REASON_XRSTORS]                 = handle_xrstors,
    [EXIT_REASON_PML_FULL]            = handle_pml_full,
    [EXIT_REASON_PREEMPTION_TIMER]        = handle_preemption_timer,
};
```

Security

How to escape from a virtualizing environment?

find bugs in the code host OS executes

1. Virtualizing hardware
2. Binary Translation

Get a qemu cve

Review the souce code or simple fuzz.

PCI device IO:

```
#include <sys/io.h>
iopl(3)//赋予当前程序读写IO端口的权限,也可以使用IOperm()来临时启用
//读取
inb(port);//byte
inw(port);//word=2 bytes
inl(port);//dwords=4 bytes
//写入
outb(val,port);
outw(val,port);
outl(val,port);
```

```
#include <asm/io.h>
#include <linux/ioport.h>

long addr=ioremap(ioaddr,iomemsize);
readb(addr);
readw(addr);
readl(addr);
readq(addr);//qwords=8 btyes

writeb(val,addr);
writew(val,addr);
writel(val,addr);
writeq(val,addr);
iounmap(addr);
```

A simple bug:CVE-2017-8380

qemu scsi device : Megasas,leak memory

```
case MFI_SEQ:
    trace_megasas_mmio_writel("MFI_SEQ", val);
    /* Magic sequence to start ADP reset */
    if (adp_reset_seq[s->adp_reset] == val) {
        s->adp_reset++;
    } else {
        s->diag = 0;
    }
    if (s->adp_reset == 6) {
        s->diag = MFI_DIAG_WRITE_ENABLE;
    }
    break;
```

```
static int adp_reset_seq[] = {0x00, 0x04, 0x0b, 0x02, 0x07, 0x0d};
```

```
case MFI_DIAG:
    retval = s->diag;
    trace_megasas_mmio_readl("MFI_DIAG", retval);
    break;
case MFI_OSP1:
```

Try some function that can not get easily

qemu-system-x86_64 -m 4096 --enable-kvm -device megasas,id=bus0 -drive file=16GB.img,if=none,id=d0 -device scsi-disk,drive=d0,bus=bus0.0 -hda centos_i386.img

CTF games

Reverse : Monitor

Pwn : Qemu escape,Qemu feature(0ctf finals)

THANKS