ISA 673
Operating Systems’ Security

Virtual Machine Monitors
Virtual Machine Monitors

- Virtual Machine Monitors (VMMs) are everywhere
  - Industry commitment
    - Software: VMware, Xen, Microsoft Virtual PC
    - Hardware: Intel VT, AMD-V
    - If Intel and AMD add it to their chips, you know it’s serious...
  - Academia: lots of VMM-based projects and papers

- An old idea: developed by IBM in 60s & 70s

- Today
  - What is it, what problems have to be solved, how to solve them
  - Survey some virtualization systems
  - Briefly outline cool things you can do with virtualization
What is a VMM?

- We have seen that an OS already virtualizes
  - System calls, processes, virtual memory, file system, sockets, etc.
  - Applications program to this interface
- A VMM virtualizes an entire physical machine
  - Interface supported is the hardware
    - OS defines a higher-level interface
  - VMM provides the **illusion** that software has full control over the hardware (of course, VMM is in control)
  - VMM “applications” run in virtual machines (c.f., OS processes)
What is a VMM? (Cont.)

- Implications
  - You can boot an operating system in a virtual machine
  - Run multiple instances of an OS on same physical machine
  - Run different OSes simultaneously on the same machine
    - Linux on Windows, Windows on Mac, etc.

- Capabilities
  - Virtualize Resources and Isolate processes
  - Share and meter resources
  - Provide “machine in machine” types of abstraction
  - How many levels can we really go down?

VMM stands for Virtual Machine Monitor & VM for Virtual Machine
Why is that important?

- **Resource utilization**
  - Machines today are powerful, want to multiplex their hardware
    - e.g., ISP hosting can divvy up a physical machine to customers
  - Can migrate VMs from one machine to another without shutdown

- **Software use and development**
  - Can run multiple OSes simultaneously
    - No need to dual boot
  - Can do system (e.g., OS) development at user-level

- **Security**
  - Securing a Browser using virtualization (Demo).
  - What are the benefits and problems?
Why is that important? (Cont.)

- Common theme is manipulating applications/services at the granularity of a machine
  - Specific version of OS, libraries, applications, etc., as package

- The importance here is to understand that there are many types of virtualization:
  - Full Emulated Virtualization (Super Heavy Weight)
  - Full Partly-Emulated Virtualization (Heavy Weight)
  - Para-Virtualization (Medium Weight)
  - OS-Level and Application Virtualization (Very Light)
Types of Virtualization

- **Emulation**
  - VM emulates/simulates complete hardware
  - Unmodified guest OS for a different PC can be run
  - Bochs, VirtualPC for Mac, QEMU, Virtualbox, Vmware Workstation

- **Full/native Virtualization**
  - VM simulates “enough” hardware to allow an unmodified guest OS to be run in isolation
  - Same hardware CPU
    - IBM VM family, VMWare Workstation, Parallels,…
Para-virtualization

- VM does not simulate hardware
- Use special API that a modified guest OS must use
- Hypercalls trapped by the Hypervisor and serviced
  - Xen, VMWare ESX Server
Types of Virtualization (Cont.)

- **OS-level virtualization**
  - OS allows multiple secure virtual servers to be run
  - Guest OS is the same as the host OS, but appears isolated
  - apps see an isolated OS
    - Solaris Containers, BSD Jails, Linux Vserver

- **Application level virtualization**
  - Application is gives its own copy of components that are not shared (E.g., own registry files, global objects)
  - VE prevents conflicts
    - Java Virtual Machine, Interpreted languages
Full vs Lightweight Virtualization

Lightweight Virtualization

Full Virtualization
VMM Requirements

- **Fidelity**
  - OSes and applications work the same without modification
    - (although we may modify the OS a bit)

- **Isolation**
  - VMM protects resources and VMs from each other

- **Performance**
  - VMM is another layer of software…and therefore overhead
    - As with OS, want to minimize this overhead
  - VMware:
    - CPU-intensive apps: 2-10% overhead
    - I/O-intensive apps: 25-60% overhead
CPU Virtualization

VMM needs to multiplex VMs on CPU

- How? Just as you would expect
  - Timeslice the VMs
  - Each VM will timeslice its OS/applications during its quantum

- Typically relatively simple scheduler
  - Round robin, work-conserving (give unused quantum to other VMs)
Virtualizing Events & I/O

- VMM receives interrupts, exceptions
  - Needs to vector to appropriate VM
  - Craft appropriate handler invocation, emulate event registers

- OSes can no longer interact directly with I/O devices
  - VMWare Workstation: generic devices only (hosted)
    - E.g., AMD Lance chipset/PCNet Ethernet device
    - Load driver into OS in VM, OS uses it normally
    - Driver knows about VMM, cooperates to pass the buck to a real device driver (e.g., on underlying host OS)
  - VMware ESX Server: drivers run in VMM (hypervisor)
Virtualized I/O Models

Abramson et al., “Intel Virtualization Technology for Directed I/O”, Intel Technology Journal, 10(3) 2006
Virtualization in Mobile Devices

- What are the problems?
  - Why did Google use the Dalvik VM?
  - Is it enough in terms of application Security?
- Can we monitor operations inside Dalvik?
  - Who will guard the guarding program?
    - No mechanism for two-level control
    - Linux-level processes not visible
- There is a need for higher level control
  - See the paper by Gernot Heiser
  - Discussion
Summary

- VMMs multiplex virtual machines on hardware
  - Export the hardware interface
  - Run OSes in VMs, apps in OSes unmodified
  - Run different versions, kinds of OSes simultaneously

- Intel and AMD are adding virtualization support
  - Goal is to fully virtualize architecture
  - Transparent trap-and-emulate approach now feasible
  - Echoes hardware support originally implemented by IBM

- Lesson: Never underestimate the power of indirection