Virtual Machines
What is a Virtual Machine?

- Java Virtual Machine (Application Virtualization)
- Software to simulate hardware
- Independent
- ‘Separate’
- Use one piece of hardware to simulate many computers
Topics

- What are Virtual Machines?
- Why are they related to security?
- Getting past Virtual Machines
Flavors of Virtualization

- Hardware Virtual Machine
  - Emulation (full system virtualization)
    - Complete hardware virtualization. Unmodified Guest OS for a different CPU can run
  - Native Virtualization (full virtualization)
    - Simulates hardware to run an unmodified OS, but OS has to be for the same type of CPU
Application Virtual Machine (Paravirtualization)
- Does not simulate hardware, but offers API that requires OS modifications like JIT compilers or interpreters

Virtual Environment (Virtual Private Server)
- Used to run applications, doesn’t simulate a kernel
- Operating System-Level Virtualization
Machine Aggregation (clustering)

- Use number of different computers to simulate a more powerful single machine
- Parallell Virtual Machine (PVM)
- Message Parsing Interface (MPI)
Why use it?

- Running multiple operating systems
- Physical space
- Mobility (USB Drives)
- Sandboxing
- Honeypots
Hypervisor / VMM

- Platform allowing multiple operating systems to run
- Abstraction layer for a virtual machine
  - Equivalence
  - Resource Control
  - Efficiency
Virtualization Requirements

- Popek and Goldberg Virtualization Requirements

- Instruction Set Architecture must possess:
  - Operate in user mode or system mode
  - Uniformly addressable memory (relative to a register)

- All instructions affecting the functioning of the VMM are controlled by the VMM
A computer is virtualizable if it is virtualizable or a VMM without timing dependencies can be constructed for it. (Recursive)

x86 processors compliant:
- Intel Virtualization Technology (IVT)
  - Most P4, Pentium D, Xeon, Core Duo, Core 2 Duo
- AMD Virtualiation (AMD-V, Pacifica)
  - K8, all F’s and onwards
Explanation of the Diagram

- Kernels manage CPU, Memory and devices and interfaces them with applications
- VMM splits the left and right side to keep them isolated
- Ring level determines the amount of power that layer has
Sandboxing

- Installing new infrastructure software
- Installing downloaded software on the net
- Browsing Security – Undo Disk in VPC
Honeypots

- Used to detect malicious users
- Set up a VM network
- Let someone attack your system, then watch them, since no useful information is being stolen
- Only software layer being attacked
Misconceptions

- Virtual Machines aren’t an end-all security guarantee
- Software still using CPU and memory of host machine
- Equivalence, Resource Control and Efficiency aren’t always completely achieved
Detecting a VM

- Run loops on remote machine suspected to be a VM
- Loops contain commands a certain VM (Xen, VMWare) don’t do particularly well
- Run Loop they do well
- Detect differences of speed opposed to non-VM’s
Java Virtual Machine Attack

- Attacking a JVM that permits untrusted code to execute after it’s verified to be type-safe
- Sending JVM a program and waiting for a memory error
- Once it type-checks, it rearranges the memory so the type system is defeated
The program

- **Class A** {
  A a1;
  A a2;
  B b;
  A a4;
  A a5;
  int i;
  A a7;
}

- **Class B** {
  A a1;
  A a2;
  A a3;
  A a4;
  A a5;
  A a6;
  A a7;
}
Memory Error

- The $i^{th}$ bit of a word is flipped for some reason
- If $2^i$ is larger than the object size, $x$ xor $2^i$ is likely to point to the base of a B object.
- Then, there is an object with type A that actually points to a B object
Exploiting the Memory Error

A p;
B q;
int offst = 6*4
void write(int addr, int value) {
    p.i = addr - offst;
    q.a6.i = value;
}

- offst is the offset of the field i from the object A
- i and a6 of object B are equal offsets from their bases
- If p and q are at the same address, the second statement writes at the offset of an offset
- value is written at offst + (addr – offst) = addr
Results

- This lets anyone calling `write()` to write value `v` into address `a`
- Fill an array with machine code
- Overwrite a virtual method table with the address of the array
- Remote code execution
VMware Attack

- NAT in VMware was not validating PORT and EPRT commands from FTP
- Specially formatted commands allowed heap-based buffer overflow
- Vulnerability allowed attacker to execute code on host machine