Isolation
Virtual Machines
Covert Channels
Overview

• The confinement problem
• Isolating entities
  – Virtual machines
  – Sandboxes
• Covert channels
  – Detecting them
  – Analyzing them
  – Mitigating them
"Isolation"

• Process cannot communicate with any other process
• Process cannot be observed

Impossible for this process to leak information
  – Not practical as process uses observable resources such as CPU, secondary storage, networks, etc.
Rule of Transitive Confinement

- If $p$ is confined to prevent leaking, and it invokes $q$, then $q$ must be similarly confined to prevent leaking.
- Rule: if a confined process invokes a second process, the second process must be as confined as the first.
Lipner’s Observation (1975)

• All processes can obtain rough idea of time
  – Read system clock or wall clock time
  – Determine number of instructions executed

• All processes can manipulate time
  – Wait some interval of wall clock time
  – Execute a set number of instructions, then block
Kocher’s Attack

• This computes $x = a^{z} \mod n$, where $z = z_0 \ldots z_{k-1}$

\[
x := 1; \quad \text{atmp} := a;
\]
\[
\text{for } i := 0 \text{ to } k-1 \text{ do begin}
\]
\[
\quad \text{if } z_i = 1 \text{ then}
\]
\[
\quad \quad x := (x \times \text{atmp}) \mod n;
\]
\[
\quad \quad \text{atmp} := (\text{atmp} \times \text{atmp}) \mod n;
\]
\[
\text{end}
\]
\[
\text{result} := x;
\]

• Length of run time related to number of 1 bits in $z$
Isolation

• Virtual machines
  – Emulate computer
  – “Guest” entity cannot access underlying computer system

• Sandboxing
  – Does not emulate computer
  – Alters interface between computer, process
Virtualization Defined
For those more visually inclined...

Traditional Architecture

Virtual Architecture
Virtualization
Virtual Machine (VM)

- A program that simulates hardware of computer system
- *Virtual machine monitor* (VMM, “hypervisor”) provides VM on which conventional OS can run
  - Each VM is one subject; VMM doesn’t worry about processes running inside each VM
  - VMM mediates all interactions of VM with resources, other VMS
  - Satisfies rule of transitive closure
KVM/370

• Security-enhanced version of IBM VM/370 VMM

• Goals
  – Provide virtual machines for users
  – Prevent VMs of different security classes from communicating

• Provides minidisks; some VMs could share some areas of disk
  – Security policy controlled access to shared areas to limit communications to those allowed by policy
DEC VAX VMM

- VMM is security kernel
  - Can run Ultrix or VMS
- Invoked on trap to execute privileged instruction
  - Only VMM can access hardware directly
  - VM kernel, executive levels both mapped into physical executive level
- VMM subjects: users, VMs
  - Each VM has own disk areas, file systems
  - Each subject, object has multilevel security, integrity labels
Oracle VirtualBox

You are seeing these slides inside a VirtualBox VM 😊

Here’s a demo ...
Sandbox

- Environment in which actions of process are restricted according to security policy
  - Can add extra security-checking mechanisms to libraries, kernel
    - Program to be executed is not altered
  - Can modify program or process to be executed
    - Similar to debuggers, profilers that add breakpoints
    - Add code to do extra checks (memory access, etc.) as program runs (*software fault isolation*)
Example: Limiting Execution

• Sidewinder
  – Uses type enforcement to confine processes
  – Sandbox built into kernel; site cannot alter it

• Java VM
  – Restricts set of files that applet can access and hosts to which applet can connect

• DTE, type enforcement mechanism for DTEL
  – Kernel modifications enable system administrators to configure sandboxes
Example: Trapping System Calls

• Sandboxie (! download and use it !)
  – File system sandbox

Here’s a demo ...
Example: Trapping System Calls

- Janus: execution environment
  - Users restrict objects, modes of access
  - Two components
    - *Framework* does run-time checking
    - *Modules* determine which accesses allowed
  - Configuration file controls modules loaded, constraints to be enforced
Janus Configuration File

# basic module
basic
    # Load basic module environment variables
putenv IFS="\t\n" PATH=/sbin:/bin:/usr/bin TZ=PST8PDT
    # Define environmental variables for process
    # deny access to everything except files under /usr
path deny read,write *
    # Deny all file accesses except to those under /usr
path allow read,write /usr/*
    # allow subprocess to read files in library directories
    # needed for dynamic loading
path allow read /lib/* /usr/lib/* /usr/local/lib/*
    # Allow reading of files in these directories (all dynamic load libraries are here)
    # needed so child can execute programs
path allow read,exec /sbin/* /bin/* /usr/bin/*
    # Allow reading, execution of subprograms in these directories
Janus Implementation

- System calls to be monitored defined in modules
- On system call, Janus framework invoked
  - Validates system call with those specific parameters are allowed
  - If not, sets process environment to indicate call failed
  - If okay, framework gives control back to process; on return, framework invoked to update state
- Example: reading MIME mail
  - Embed “delete file” in Postscript attachment
  - Set Janus to disallow Postscript engine access to files
Covert Channel

• Channel using *shared* resources as a communication path

• *Covert storage channel* uses attribute of shared resource

• *Covert timing channel* uses temporal or ordering relationship among accesses to shared resource
Example: File Manipulation

• Communications protocol:
  – $p$ sends a bit by creating a file called 0 or 1, then a second file called $send$
    • $p$ waits until $send$ is deleted before repeating to send another bit
  – $q$ waits until file $send$ exists, then looks for file 0 or 1; whichever exists is the bit
    • $q$ then deletes 0, 1, and $send$ and waits until $send$ is recreated before repeating to read another bit

• Covert storage channel: resource is directory, names of files in directory
Example: Using Real Time Clock

- KVM/370 had covert timing channel
  - VM1 wants to send 1 bit to VM2
  - To send 0 bit: VM1 relinquishes CPU as soon as it gets CPU
  - To send 1 bit: VM1 uses CPU for full quantum
  - VM2 determines which bit is sent by seeing how quickly it gets CPU
  - Shared resource is CPU, timing because real-time clock used to measure intervals between accesses
Example: Ordering of Events

- Two VMs
  - Share cylinders 100–200 on a disk
  - One is *High*, one is *Low*; process on *High* VM wants to send to process on *Low* VM
- Disk scheduler uses SCAN algorithm
- *Low* process seeks to cylinder 150 and relinquishes CPU
  - Now we know where the disk head is
Example: Ordering (continued)

- *High* wants to send a bit
  - To send 1 bit, *High* seeks to cylinder 140 and relinquish CPU
  - To send 0 bit, *High* seeks to cylinder 160 and relinquish CPU
- *Low* issues requests for tracks 139 and 161
  - Seek to 139 first indicates a 1 bit
  - Seek to 161 first indicates a 0 bit
- Covert timing channel: uses ordering relationship among accesses to transmit information
Noise

- **Noiseless covert channel** uses shared resource available *exclusively* to sender and receiver

- **Noisy covert channel** uses shared resource available to sender, receive, and others
  - Need to minimize interference enough so that message can be read in spite of others’ use of channel
Key Properties

• Existence
  – Determining whether the covert channel exists

• Bandwidth
  – Determining how much information can be sent over the channel
How do we detect them?

- Covert channels require sharing
- Manner of sharing controls which subjects can send, which subjects can receive information using that shared resource
- Porras, Kemmerer: model flow of information through shared resources with a tree
  - Called covert flow trees *(study them in more advanced class)*
Mitigation

• Goal: obscure amount of resources a process uses
  – Receiver cannot determine what part sender is using and what part is obfuscated

• How to do this?
  – Devote uniform, fixed amount of resources to each process
  – Inject randomness into allocation, use of resources
Key Points

• Confinement problem: prevent leakage of information
  – Solution: separation and/or isolation
• Shared resources offer paths along which information can be transferred
• Covert channels difficult if not impossible to eliminate
  – Bandwidth can be greatly reduced, however!