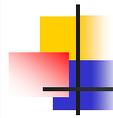


Confinement Problem

- The confinement problem
- Isolating entities
 - Virtual machines
 - Sandboxes
- Covert channels
 - Mitigation

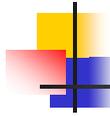
1



Example Problem

- Server balances bank accounts for clients
- Server security issues:
 - Record *correctly* who used it
 - Send *only* balancing info to client
- Client security issues:
 - Log use *correctly*
 - Do not save or retransmit data client sends

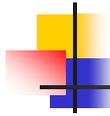
2



Generalization

- Client sends request, data to server
- Server performs some function on data
- Server returns result to client
- Access controls:
 - Server must ensure the resources it accesses on behalf of client include *only* resources client is authorized to access
 - Server must ensure it does not reveal client's data to any entity not authorized to see the client's data

3

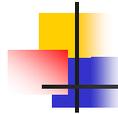


Confinement Problem

(Lampson 1973)

- Problem of preventing a server from leaking information that the user of the service considers confidential

4



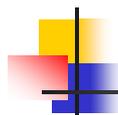
Total Isolation

- Process cannot communicate with any other process and cannot be observed

Impossible for this process to leak information

- Not practical as process uses observable resources such as CPU, secondary storage, networks, etc.

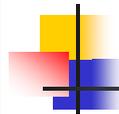
5



Lipner's observation

- 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

6



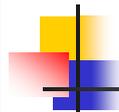
Kocher's Attack

- This computes $x = a^z \bmod n$, where $z = z_0 \dots z_{k-1}$

```
x := 1; atmp := a;
for i := 0 to k-1 do begin
  if zi = 1 then
    x := (x * atmp) mod n;
    atmp := (atmp * atmp) mod n;
end
result := x;
```

- Length of run time related to number of 1 bits in z

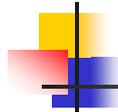
7



2 approaches to isolation

- Virtual machines
 - Emulate computer
 - Process cannot access underlying computer system and anything not part of that system
- Sandboxing
 - Does not emulate computer
 - Alters interface between computer and process

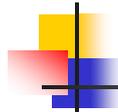
8



Virtual Machine (VM)

- A program that simulates hardware of computer system
- *Virtual machine monitor* (VMM) provides VM on which conventional OS can run (without modifications)
 - Each VM is one subject; VMM knows nothing about processes running on each VM
 - VMM mediates all interactions of VM with resources and other VMS
 - Satisfies rule of transitive closure

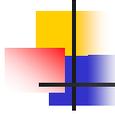
9



Example: KVM/370

- Security-enhanced version of IBM VM/370 Virtual Machine Monitor
- 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

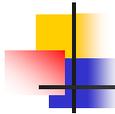
10



Sandbox

- Environment in which actions of process are restricted according to security policy
- Enforced in one of two ways:
 - Add extra security-checking mechanisms to libraries or kernel
 - Program to be executed is not altered
 - 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*)

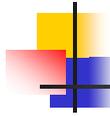
11



Example: Limiting Execution

- (operational kernel of) Sidewinder firewall
 - 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

12



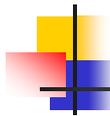
Definition of Covert Channels

- Is a **path of communication**, a mechanism that can be used in an *unexpected* manner to transfer data to an unauthorized subject
- Overt channels block information by using authorization and access controls
- Covert channels are harder: it has been argued that it is *impossible* to remove covert channels
 - *Something* is always shared (see non-interference)
 - because they are *unexpected*, there is no access control

Note: a covert channel does not have to be a 'good' channel

- Attacker can be patient, leak information slowly

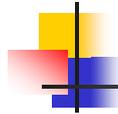
13



Example: File Manipulation

- Processes p , q not allowed to communicate
 - But they share a file system (!)
- 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

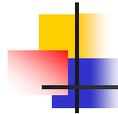
14



Example: 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

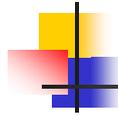
15



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 issues read request to cylinder 150 and relinquishes CPU
 - Now we know where the disk head is

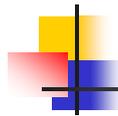
16



Example (*con't*)

- *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

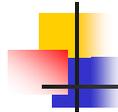
17



Key Properties

- Existence
 - Determining whether the covert channel exists
- Bandwidth
 - Determining how much information can be sent over the channel

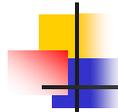
18



Covert Channel Analysis

- Could argue that it is *impossible* to remove covert channels
 - *Something* is always shared
- Exploiting a covert channel is an *inside* job
 - Inside *people* can access the "talk-net", so covert channel analysis is useless against them
 - Inside *code* : this is an *integrity* problem, so the problem is to keep hostile code out of the system

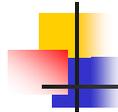
19



Coping with covert channels

- After identification
 - close the channel
 - slow it down by introducing noise
 - for timing channel - hide the exact timing of events
 - tolerate it
 - estimate the bandwidth (the amount of information transmitted per unit of time)
 - audit occurrence of events involved in usage of the channel
 - cost/benefit tradeoff
 - Example
 - Ship location classified until next commercial satellite flies overhead
 - Can covert channel transmit location before this?

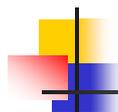
20



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

21



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

22