ECS 289M Lecture 21

May 19, 2006

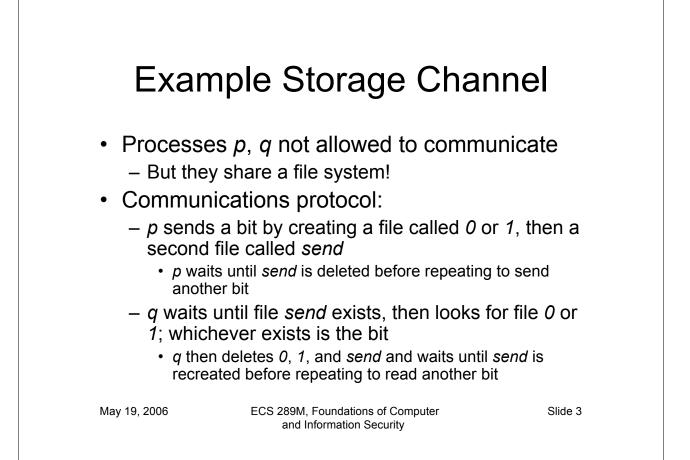
Covert Channels

- Shared resources as communication paths
- Covert storage channel uses attribute of shared resource

– Disk space, message size, etc.

• *Covert timing channel* uses temporal or ordering relationship among accesses to shared resource

- Regulating CPU usage, order of reads on disk



Example Timing Channel

- System has two VMs
 - Sending machine S, receiving machine R

• To send:

- For 0, S immediately relinquishes CPU
 - For example, run a process that instantly blocks
- For 1, S uses full quantum
 - For example, run a CPU-intensive process
- R measures how quickly it gets CPU
 - Uses real-time clock to measure intervals between access to shared resource (CPU)

Example Covert Channel

- Uses ordering of events; does not use clock
- Two VMs sharing disk cylinders 100 to 200
 - SCAN algorithm schedules disk accesses
 - One VM is High (H), other is Low(L)
- Idea: L will issue requests for blocks on cylinders 139 and 161 to be read
 - If read as 139, then 161, it's a 1 bit
 - If read as 161, then 139, it's a 0 bit

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 5

How It Works

- L issues read for data on cylinder 150
 Relinguishes CPU when done; arm now at 150
- *H* runs, issues read for data on cylinder 140 – Relinguishes CPU when done; arm now at 140
- *L* runs, issues read for data on cylinders 139 and 161

- Due to SCAN, reads 139 first, then 161

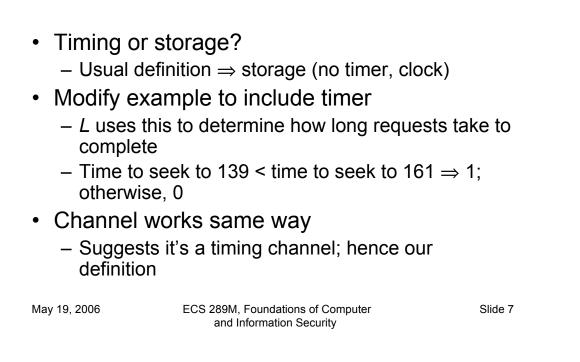
- This corresponds to a 1

 To send a 0, H would have issued read for data on cylinder 160

May 19, 2006

ECS 289M, Foundations of Computer and Information Security

Analysis



Noisy vs. Noiseless

- Noiseless: covert channel uses resource available only to sender, receiver
- Noisy: covert channel uses resource available to others as well as to sender, receiver
 - Idea is that others can contribute extraneous information that receiver must filter out to "read" sender's communication

ECS 289M, Foundations of Computer and Information Security

Key Properties

- *Existence*: the covert channel can be used to send/receive information
- *Bandwidth*: the rate at which information can be sent along the channel
- Goal of analysis: establish these properties for each channel
 - If you can eliminate the channel, great!
 - If not, reduce bandwidth as much as possible

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 9

Step #1: Detection

- Manner in which resource is shared controls who can send, receive using that resource
 - Noninterference
 - Shared Resource Matrix Methodology
 - Information flow analysis
 - Covert flow trees

Noninterference

- View "read", "write" as instances of information transfer
- Then two processes can communicate if information can be transferred between them, even in the absence of a direct communication path
 - A covert channel
 - Also sounds like interference ...

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 11

Example: SAT

• Secure Ada Target, multilevel security policy

• Approach:

- $\pi(i, I)$ removes all instructions issued by subjects dominated by level *I* from instruction stream *i*
- $-A(i, \sigma)$ state resulting from execution of *i* on state σ
- $-\sigma$.*v*(*s*) describes subject *s*'s view of state σ
- System is noninterference-secure iff for all instruction sequences *i*, subjects *s* with security level *l*(*s*), states σ,

 $A(\pi(i, l(s)), \sigma).v(s) = A(i, \sigma).v(s)$

May 19, 2006

ECS 289M, Foundations of Computer and Information Security

Theorem

- Version of the Unwinding Theorem
- Let Σ be set of system states. A specification is noninterference-secure if, for each subject s at security level *l*(s), there exists an equivalence relation =: Σ×Σ such that
 - for $\sigma_1, \sigma_2 \in \Sigma$, when $\sigma_1 \equiv \sigma_2, \sigma_1.v(s) = \sigma_2.v(s)$
 - for $\sigma_1, \sigma_2 \in \Sigma$ and any instruction *i*, when $\sigma_1 \equiv \sigma_2$, $A(i, \sigma_1) \equiv A(i, \sigma_2)$
 - for $\sigma \in \Sigma$ and instruction stream *i*, if $\pi(i, l(s))$ is empty, $A(\pi(i, l(s)), \sigma) \cdot v(s) = \sigma \cdot v(s)$

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 13

Intuition

- System is noninterference-secure if:
 - Equivalent states have the same view for each subject
 - View remains unchanged if any instruction is executed
 - Instructions from higher-level subjects do not affect the state from the viewpoint of the lower-level subjects

Analysis of SAT Focus on object creation instruction and readable object set In these specifications:

- -s subject with security level l(s)
- -o object with security level l(o), type $\tau(o)$
- $-\sigma$ current state
- Set of existing objects listed in a global object table $T(\sigma)$

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 15

Specification 1

• object_create:

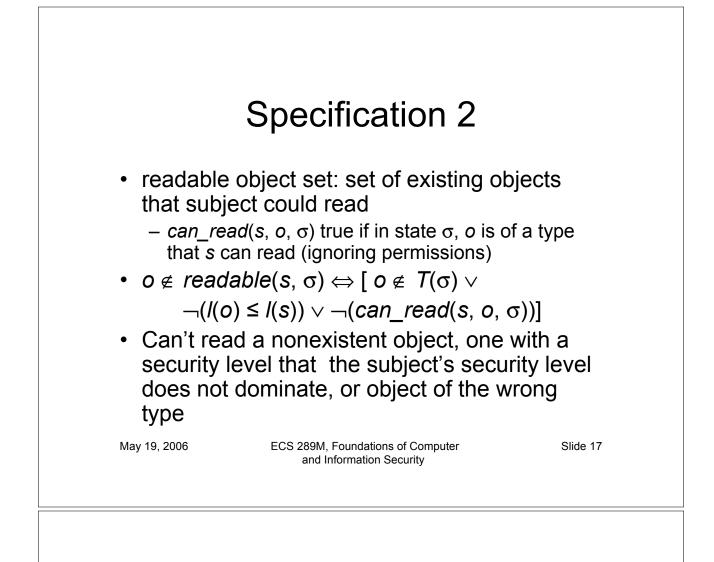
 $[\sigma' = object_create(s,o,l(o),\tau(o),\sigma) \land \sigma' \neq \sigma]$

$$(o \notin T(\sigma) \land I(s) \leq I(o)]$$

- The create succeeds if, and only if, the object does not yet exist and the clearance of the object will dominate the clearance of its creator
 - In accord with the "writes up okay" idea

May 19, 2006

ECS 289M, Foundations of Computer and Information Security



Specification 3

- SAT enforces tranquility
 - Adding object to readable set means creating new object
- Add to readable set:
 - $\begin{array}{l} [o \notin readable(s, \sigma) \land o \in readable(s, \sigma')] \Leftrightarrow [\sigma' = \\ object_create(s, o, l(o), \tau(o), \sigma) \land o \notin T(\sigma) \land l(s') \leq \\ l(o) \leq l(s) \land can_read(s, o, \sigma')] \end{array}$
- Says object must be created, levels and discretionary access controls set properly

May 19, 2006

ECS 289M, Foundations of Computer and Information Security

Check for Covert Channels

- σ_1 , σ_2 the same except:
 - o exists only in latter
 - $-\neg(l(o) \leq l(s))$
- Specification 2:
 - $-o \notin readable(s, \sigma_1) \{ o \text{ doesn't exist in } \sigma_1 \}$
 - $-o \notin readable(s, \sigma_2) \{ \neg (l(o) \leq l(s)) \}$
- Thus $\sigma_1 \equiv \sigma_2$ – Condition 1 of theorem holds

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 19

Continue Analysis

- s' issues command to create o with:
 - l(o) = l(s)
 - of type with *can_read*(s, o, σ_1')
 - σ_1' state after *object_create*(s', o, *l*(o), τ (o), σ_1)
- Specification 1
 - $-\sigma_1$ differs from σ_1 with *o* in $T(\sigma_1)$
- · New entry satisfies:
 - can_read(s, o, σ_1')
 - $I(s') \le I(o) \le I(s)$, where s' created o

May 19, 2006

ECS 289M, Foundations of Computer and Information Security

Continue Analysis

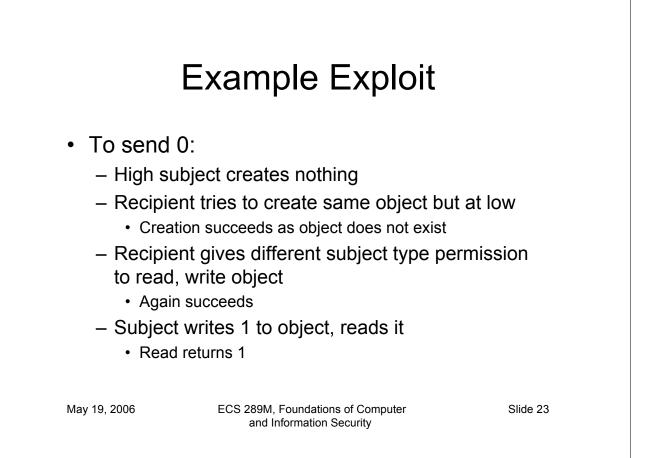
- o exists in σ_2 so: $\sigma_2' = object_create(s', o, \sigma_2) = \sigma_2$
- But this means
 - $\neg [A(object_create(s', o, l(o), \tau(o), \sigma_2), \sigma_2) \equiv A(object_create(s', o, l(o), \tau(o), \sigma_1), \sigma_1)]$
 - Because create fails in σ_2 but succeeds in σ_1
- So condition 2 of theorem fails
- This implies a covert channel as system is not noninterference-secure

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 21

Example Exploit

- To send 1:
 - High subject creates high object
 - Recipient tries to create same object but at low
 - Creation fails, but no indication given
 - Recipient gives different subject type permission to read, write object
 - Again fails, but no indication given
 - Subject writes 1 to object, reads it
 - Read returns nothing





- Can analyze covert storage channels
 - Noninterference techniques reason in terms of security levels (attributes of objects)
- Covert timing channels much harder
 - You would have to make ordering an attribute of the objects in some way

SRMM

- Shared Resource Matrix Methodology
- Goal: identify shared channels, how they are shared
- Steps:
 - Identify all shared resources, their visible attributes [rows]
 - Determine operations that reference (read), modify (write) resource [columns]
 - Contents of matrix show how operation accesses the resource

May 19, 2006

ECS 289M, Foundations of Computer and Information Security Slide 25

Example

- Multilevel security model
- File attributes:
 - existence, owner, label, size
- File manipulation operations:
 - read, write, delete, create
 - create succeeds if file does not exist; gets creator as owner, creator's label
 - others require file exists, appropriate labels
- · Subjects:
 - High, Low

May 19, 2006

ECS 289M, Foundations of Computer and Information Security

Shared Resource Matrix

	read	write	delete	create
existence	R	R	R, M	R, M
owner			R	М
label	R	R	R	М
size	R	М	М	М
May 19, 2006 ECS 289M, Foundations of Computer Slide 27				

Covert Storage Channel

and Information Security

- Properties that must hold for covert storage channel:
 - 1. Sending, receiving processes have access to same *attribute* of shared object;
 - 2. Sender can modify that attribute;
 - 3. Receiver can reference that attribute; and
 - 4. Mechanism for starting processes, properly sequencing their accesses to resource

Example

- Consider attributes with both R, M in rows
- · Let High be sender, Low receiver
- create operation both references, modifies existence attribute
 - Low can use this due to semantics of create
- Need to arrange for proper sequencing accesses to existence attribute of file (shared resource)

May 19, 2006

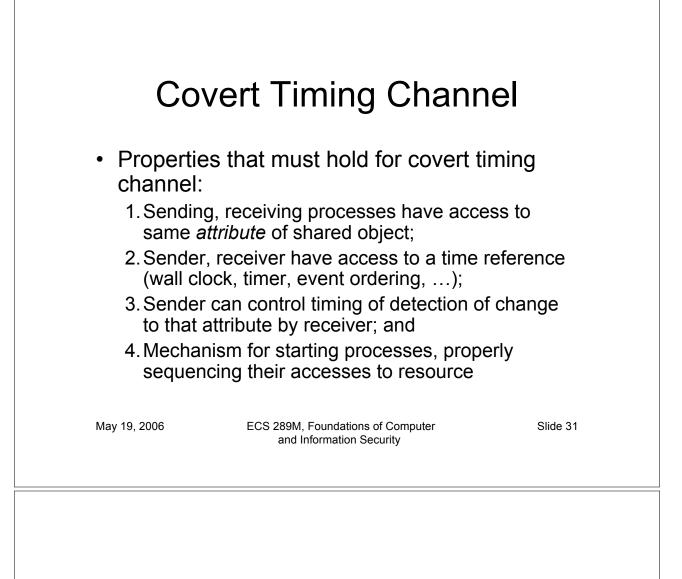
ECS 289M, Foundations of Computer and Information Security Slide 29

Use of Channel

- 3 files: ready, done, 1bit
- Low creates *ready* at High level
- High checks that file exists
 - If so, to send 1, it creates 1bit; to send 0, skip
 - Delete ready, create done at High level
- Low tries to create *done* at High level
 - On failure, High is done
 - Low tries to create 1bit at level High
- Low deletes *done*, creates *ready* at High level

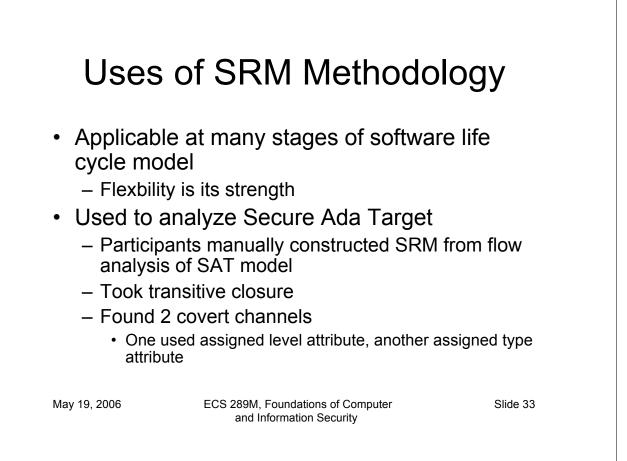
May 19, 2006

ECS 289M, Foundations of Computer and Information Security



Example

- Revisit variant of KVM/370 channel
 - Sender, receiver can access ordering of requests by disk arm scheduler (attribute)
 - Sender, receiver have access to the ordering of the requests (time reference)
 - High can control ordering of requests of Low process by issuing cylinder numbers to position arm appropriately (timing of detection of change)
 - So whether channel can be exploited depends on whether there is a mechanism to (1) start sender, receiver and (2) sequence requests as desired



Summary

- Methodology comprehensive but incomplete
 - How to identify shared resources?
 - What operations access them and how?
- Incompleteness a benefit
 - Allows use at different stages of software engineering life cycle
- Incompleteness a problem
 - Makes use of methodology sensitive to particular stage of software development